Modeling Distributed Product Development Processes in Small and Medium Enterprises

by

Armen A. Mkrtchyan

B.S. Electrical Engineering, University of North Dakota, 2009
S.M. Aeronautics and Astronautics, Massachusetts Institute of Technology, 2011

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2015

©2015 Armen A. Mkrtchyan. All rights reserved.

The author hereby grants to MIT permission to reproduce and to distribute publicly paper and electronic copies of this thesis document in whole or in part in any medium now known or hereafter created.

Signature redacted

Author ..............................................................................................................

Armen Mkrtchyan, Department of Aeronautics and Astronautics

May 21, 2015

Certified by ..............................................................................................

Deborah Nightingale

Professor of the Practice, Aeronautics and Astronautics

Committee Chair

Certified by ..............................................................................................

Jayakanth Srinivasan

Research Scientist, Sloan School of Management

Thesis Supervisor

Certified by ..............................................................................................

Amos Winter

Associate Professor, Mechanical Engineering

Thesis Committee Member

Accepted by ..............................................................................................

Paulo C. Lozano

Associate Professor, Aeronautics and Astronautics

Chair, Committee on Graduate Students
DISCLAIMER NOTICE

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available.

Thank you.

Some pages in the original document contain text that is illegible.

Pages 99-100
Modeling Distributed Product Development Processes in Small and Medium Enterprises

by

Armen A. Mkrtchyan

Submitted to the Department of Aeronautics and Astronautics on May 21st, 2015 in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Aeronautics and Astronautics.

Abstract

Effective and efficient product development (PD) is critical to the success of many firms. The market’s emphasis on getting faster and cheaper products has forced firms to shift their PD practices from sequential to distributed development practices. This shift has significantly increased complexity and dynamism of PD processes, but often is not sufficiently accounted for by PD managers. Transitioning to distributed PD processes is even more challenging for Small and Medium Enterprises (SMEs), who usually lack resources to secure external help. Furthermore, SMEs have been increasingly using distributed PD to not only cut costs but also accelerate PD processes, which adds extra complexity to the management of PD processes.

The objective of this thesis is to develop a generalizable model of distributed PD to aid SMEs better manage their PD processes. First, I identified the necessary attributes that have to be captured in distributed PD process modeling within SMEs using firm interviews and academic literature. I validated the attribute list using firm surveys. Next, I developed a discrete-event simulation model that accounted for task structure, rework, multiple teams, flexible work hours, individual performance ratings and learning curve factors. The model is designed to calculate project completion time and cost. In addition, it generates a probability distribution function of completion time that can be used to further guide PD manager decision-making. I encapsulated the model within a software application called SimLink™ for both Mac OS® and iOS® mobile devices.

Finally, I adopted a multistage validation process using both historical data sets and ongoing project data to accurately replicate observed results and guide decision making in real world scenario. I built additional confidence in model’s predictive ability through sensitivity and stochasticity analysis. The application can be used to evaluate the impact of different PD attributes and analyze various PD scenarios before implementing them in real world. Specifically, the application (and underlying model) has been used to investigate cost and schedule targets for various PD staffing configurations, as well as to analyze the impact of around-the-clock development activities. Lastly, model limitations and generalizability to other PD settings are discussed.

Thesis Supervisor: Jayakanth Srinivasan
Title: Research Scientist, Sloan School of Management
To

My Parents for Their Sacrifices

&

Tatevik H. for Her Patience and Encouragement
Acknowledgments

First, I would like to thank my research advisor, Jayakanth Srinivasan for being my mentor, my counselor, and my friend. JK, it is because of your mentorship and guidance that I was able to accomplish this much. Thank you for believing in me and agreeing to be my advisor when I was not sure what type of research I wanted to do for my thesis. Thank you for always finding time to talk about life and family and helping me make better decisions. I am forever grateful for the opportunity I had to work with you and learn from you. I know I have found an amazing mentor and I look forward to a lifetime of friendship.

Thank you, Debbi Nightingale for allowing me to join SSRC three years ago and giving me freedom to work on research problems that I care about. I am sure we will find more opportunities to collaborate in the future.

Amos Winter, thank you for being on my committee and for always supporting me and being a positive person. I look forward to hearing more about your endeavors and accomplishments.

Warren Seering, thank you for your invaluable feedback during my committee presentations and on my thesis draft. Also, thank you for taking the time to discuss my future career options.

Aram Hajian, thank you for your support in the last semester. Having a space at the AUA to work on my thesis was crucial for helping me finish my thesis draft on time. I look forward to working with you in Armenia.

I would also like to thank Eytan Modiano and Hamsa Balakrishnan for taking me as a TA for the Unified Engineering class. I learned a great deal about teaching and working with students.

I am also thankful to Luys Foundation for their assistance during my graduate studies. Jacqueline Karaaslanian, thank you for always supporting me in my endeavors and encouraging me to achieve more.

Also, I wish to acknowledge the Legatum Center at MIT and the Armenian General Benevolent Union for their financial support during my PhD studies.

Will Guyster, I have a great appreciation for the work you do to help the Legatum Fellows implement their ideas. Thanks for all your help with my business plan and for your friendship.
Joe Teja, I cannot thank you enough for the numerous hours you spent reviewing my business plan and helping me file a patent application. I look forward to welcoming you in Armenia.

Beth Marois, thank you for encouraging me to move forward despite numerous obstacles. Also, thanks for compiling so many letters for me, especially when I was travelling. I hope we can stay friends for a long time to come.

Mirna and Ahmed, I will miss you guys very, very much. I will miss our lunch and dinner conversations and working in SP at nights. Thanks for all the great memories and experiences. Thanks for your help preparing for my committee meetings and presentations. I loved living in the same residence with you and look forward to staying friends and visiting each other.

I also want to thank Andrew, Burcu (B), Michael P., William, Carrie, Elan, Dina, Adi, Vazrik, and Velibor for making graduate life fun and interesting.

Also, thank you, Karlen, Liana, Alik, Karo, Anna, Masis, Levon, Seda, Mkrtich, Zhirayr, Tigran, Artak, Lusine, and Shushanik for your friendship and support. I look forward to spending the summer with you.

Tatevik, you have stuck with me through difficult times and have always supported me. Thank you for believing in me and always pushing me to do my best, regardless of what I am doing. Your positive attitude, love, and encouragement are contagious and I am sure you will continue to inspire me for years to come. I look forward to many more breathtaking moments with you.

My parents and family have always been the driving force behind my accomplishments. It is impossible to describe how grateful I am for all the sacrifices that my parents endured to get me where I am today. It is due to their determination, trust, and faith that I was able to push forward in difficult times. I am indebted to my mother who spent countless hours instilling in me the value of education. Thank you to my father for doing everything to provide me with the best possible opportunities. I am also thankful to my brothers, Gevorg and Vahagn for their love and support. Thank you, Anushik and Elena for bringing more joy into our lives.

Last but not least, I thank God for all the opportunities and blessings I have had in my life.
# Table of Contents

## Acknowledgments

Table of Contents .......................................................................................................................... 7

## List of Figures

List of Tables ...................................................................................................................................... 15

List of Acronyms ............................................................................................................................... 17

## 1. Introduction ................................................................................................................................. 19

### 1.1. Motivation ............................................................................................................................... 19

#### 1.1.1. Addressing the need for small and medium enterprises ......................................................... 20

#### 1.1.2. Accounting for distributed product development ................................................................. 20

#### 1.1.3. Creating a practical model / tool ............................................................................................ 21

### 1.2. Research Questions ................................................................................................................. 21

### 1.3. Thesis Outline ........................................................................................................................... 21

## 2. Identifying Modeling Attributes .................................................................................................. 25

### 2.1. Introduction ............................................................................................................................... 25

### 2.2. Research Approach .................................................................................................................. 27

#### 2.2.1. Past literature findings ........................................................................................................... 27

#### 2.2.2. Interviews and surveys ......................................................................................................... 28

#### 2.2.3. Firm data .................................................................................................................................. 29

### 2.3. Identifying PD Capabilities ....................................................................................................... 31

#### 2.3.1. Introduction ........................................................................................................................... 31

### 2.4. Synthesis .................................................................................................................................... 39

#### 2.4.1. Summary of firm capabilities .................................................................................................... 39

#### 2.4.2. Survey results ......................................................................................................................... 40

#### 2.4.3. Comparing capabilities with successful firms operating in developed countries ................. 43

### 2.5. Affected PD Attributes ............................................................................................................ 43

#### 2.5.1. Attributes impacted by flexible PD .......................................................................................... 45

#### 2.5.2. Attributes impacted by modular PD ........................................................................................ 45

#### 2.5.3. Attributes impacted by entrepreneurship focused human capital management ...................... 46

#### 2.5.4. Attributes impacted by effective and efficient customer sensemaking .................................. 47

#### 2.5.5. Attributes impacted by shared identity and goals .................................................................... 47

#### 2.5.6. Summary of PD attribute mapping .......................................................................................... 48

### 2.6. Summary .................................................................................................................................... 49

## 3. Background: Modeling Product Development Process ............................................................... 51

### 3.1. Graph Based ............................................................................................................................. 52

#### 3.1.1. Directed graph ......................................................................................................................... 52

#### 3.1.2. CPM/PERT ............................................................................................................................... 53

#### 3.1.3. GERT/Q-GERT ...................................................................................................................... 53

#### 3.1.4. SADT/IDEF ............................................................................................................................. 54

#### 3.1.5. Petri net .................................................................................................................................... 55
3.1.6. Signal flow graph .................................................................................................................. 55
3.1.7. Design roadmap ....................................................................................................................... 56
3.1.8. Precedence diagram method ................................................................................................. 56
3.1.9. Gantt chart ............................................................................................................................ 57
3.2. Matrix Based ............................................................................................................................ 58
  3.2.1. DSM ..................................................................................................................................... 58
  3.2.2. WTM ..................................................................................................................................... 59
3.3. Analytical .................................................................................................................................. 60
  3.3.1. Mathematical programming ................................................................................................. 60
  3.3.2. Queuing theory ....................................................................................................................... 61
3.4. Simulation Models .................................................................................................................... 62
  3.4.1. System dynamics ..................................................................................................................... 62
  3.4.2. Agent-based modeling ........................................................................................................... 63
  3.4.3. Discrete-event simulation ..................................................................................................... 65
3.5. Simulation Paradigm Comparison ............................................................................................... 67
3.6. Summary .................................................................................................................................... 71
4. Model Development ..................................................................................................................... 73
  4.1. Model Constructs ....................................................................................................................... 74
    4.1.1. Events ................................................................................................................................ 75
    4.1.2. Arrival processes .................................................................................................................. 75
    4.1.3. Service processes ................................................................................................................ 77
    4.1.4. Queue .................................................................................................................................. 82
  4.2. Mapping PD Attributes to Model Constructs .......................................................................... 83
    4.2.1. Task structure ....................................................................................................................... 83
    4.2.2. Team structure .................................................................................................................... 84
    4.2.3. Developer flexibility ............................................................................................................. 84
    4.2.4. Rework ............................................................................................................................... 85
    4.2.5. Task relatedness ................................................................................................................ 85
    4.2.6. Learning .............................................................................................................................. 85
    4.2.7. Coordination cost ................................................................................................................ 86
  4.3. Task Scheduling ....................................................................................................................... 87
  4.4. Model Outputs ......................................................................................................................... 88
    4.4.1. DES-based metrics .............................................................................................................. 88
    4.4.2. PD specific metrics ............................................................................................................. 89
  4.5. Summary .................................................................................................................................. 90
5. Model Verification and Validation ................................................................................................. 91
  5.1. Model Implementation ............................................................................................................. 92
    5.1.1. MATLAB® prototype .......................................................................................................... 93
    5.1.2. SimLink™ software ............................................................................................................. 96
  5.2. Model Advantages ................................................................................................................... 102
    5.2.1. Holistic .............................................................................................................................. 103
    5.2.2. Relevance to practice ......................................................................................................... 103
    5.2.3. Resource consideration ...................................................................................................... 103
Appendix C: Evaluating PDF of Triangular Distribution .................................................... 173
Appendix D: Description of SimLink™ Application .......................................................... 175
   D.1 Screenshot of MATLAB® Prototype Application ................................................... 175
   D.2 Mac OS® SimLink™ Application ......................................................................... 176
Appendix E: Support Data for Model Verification & Validation ...................................... 185
   E.1 Calculating Values for Triangular PDF ................................................................. 185
   E.2 Rework Probability and Impact Matrices for UV Project ........................................ 186
   E.3 Modeling data for Mobile Development Project .................................................... 187
   E.4 Modeling Data for Network Improvement Project .................................................. 188
   E.5 Modeling Data for New Feature Implementation Project ....................................... 191
   E.5 Additional Developer and QA Parameters .............................................................. 192
References ....................................................................................................................... 193
List of Figures

FIGURE 1: THESIS RESEARCH METHODOLOGY.................................................................23
FIGURE 2: MODELING PARADIGMS WITH CORRESPONDING ABSTRACTION LEVELS........................................68
FIGURE 3: HIGH-LEVEL REPRESENTATION OF THE MODEL......................................73
FIGURE 4: GRAPHICAL REPRESENTATION OF MEDIAN LATIN HYPERCUBE SAMPLING METHOD. 79
FIGURE 5: MODEL VALIDATION AND REFINEMENT PROCESS.....................................91
FIGURE 6: MODEL CONFIDENCE VS. VALUE / COST TRADEOFF..................................92
FIGURE 7: BLOCK DIAGRAM OF THE PROTOTYPE MODEL IMPLEMENTED IN MATLAB® ENVIRONMENT.............................................................93
FIGURE 8: WINDOW TO ADJUST SIMULATION SETTINGS.................................................97
FIGURE 9: SIMLINK™ OVERVIEW SCREEN.................................................................97
FIGURE 10: DEVELOPER WINDOW............................................................................99
FIGURE 11: MANAGE TEAM WINDOW.......................................................................100
FIGURE 12: DEVELOPER PARAMETERS.........................................................................100
FIGURE 13: TASKS MAIN WINDOW.............................................................................100
FIGURE 14: TASK PARAMETER WINDOW.....................................................................101
FIGURE 15: RESULTS WINDOW....................................................................................102
FIGURE 16: SAMPLE PROJECT TIMELINE......................................................................102
FIGURE 17: THE PDF OF A SAMPLE TRIANGULAR DISTRIBUTION WITH ITS MAIN PARAMETERS..................106
FIGURE 18: THE PDF OF PROJECT COMPLETION TIME FOR UV PROJECT..................109
FIGURE 19: ACTUAL VS. ESTIMATED DURATIONS FOR MOBILE SOFT. DEV. PROJECT......112
FIGURE 20: PART OF THE TIMELINE FOR MOBILE SOFT. DEV. PROJECT....................112
FIGURE 21: SIMULATED PDF OF PROJECT COMPLETION TIME FOR NETWORK IMPROVEMENT PROJECT...114
FIGURE 22: NEW FEATURE IMPLEMENTATION PROJECT TIME AND COST COMPARISON FOR VARIOUS SCENARIOS......116
FIGURE 23: SENSITIVITY ANALYSIS FOR TASK DURATION ESTIMATES..........................118
FIGURE 24: SENSITIVITY ANALYSIS FOR DEVELOPER PERFORMANCE RATING...........119
FIGURE 25: SENSITIVITY ANALYSIS FOR DEVELOPER LEARNING CURVE......................120
FIGURE 26: MODEL STOCHASTICITY - STANDARD DEVIATION COMPARISON................121
FIGURE 27: ESTIMATING PROJECT COMPLETION TIME FROM PDF FOR TWO INTERVALS........126
FIGURE 28: COST AND SCHEDULE TRADEOFF GRAPH.................................................127
FIGURE 29: COST AND DURATION TRADEOFF ANALYSIS FOR MOBILE DEV. PROJECT...127
FIGURE 30: EFFECT ON TASK DURATION WITHOUT (LEFT) AND WITH (RIGHT) PREEMPTION........131
FIGURE 31: MOBILE DEVELOPMENT PROJECT TIME AND COST COMPARISON FOR VARIOUS SCENARIOS OF DISTRIBUTED PD........................................................................................................132
FIGURE 32: MATLAB® SIMULATION SOFTWARE INTERFACE........................................175
FIGURE 33: MAC OS® SIMLINK™ STARTUP SCREEN................................................176
FIGURE 34: DEVELOPER VIEW OF THE SIMLINK™ APPLICATION..............................177
FIGURE 35: ADD / EDIT DEVELOPER VIEW.................................................................178
FIGURE 36: ADD TEAM VIEW.....................................................................................178
FIGURE 37: ADD / REPLACE TASKS VIEW OF THE APPLICATION..............................179
FIGURE 38: ADD TASK VIEW - GENERAL TASK PARAMETERS....................................180
FIGURE 39: ADD TASK VIEW - TASK RELATEDNESS................................................180
FIGURE 40: ADD TASK VIEW - REWORK PROBABILITIES.........................................181
FIGURE 41: ADD TASK VIEW - OVERLAP..................................................................181
FIGURE 42: MAC OS SIMLINK MODEL SETTINGS......................................................182
FIGURE 43: ADD TASK VIEW - REWORK IMPACT....................................................182
FIGURE 44: MAC OS® SIMLINK™ APPLICATION SAMPLE TIMELINE.......................183
FIGURE 45: RESULTS FOR A SAMPLE PROJECT..........................................................183
FIGURE 46: SAMPLE RESULTS USING ADVANCED LOGGING OPTION..........................184
List of Tables

TABLE 1: BRIEF DESCRIPTION OF INTERVIEWED FIRMS................................................................. 29
TABLE 2: FIRM GROWTH/DECLINE RATE CODING........................................................................... 30
TABLE 3: FIRM DESCRIPTIVE DATA.................................................................................................. 30
TABLE 4: ORGANIZATIONAL CAPABILITIES WITH ASSOCIATED KEY FACTORS............................. 33
TABLE 5: INTERVIEW FINDINGS - FACTORS ASSOCIATED WITH FLEXIBLE PD............................... 35
TABLE 6: INTERVIEW FINDINGS - FACTORS ASSOCIATED WITH MODULAR PD............................... 36
TABLE 7: INTERVIEW FINDINGS - FACTORS ASSOCIATED WITH ENTREPRENEURSHIP FOCUSED HUMAN CAPITAL MANAGEMENT.......................................................... 37
TABLE 8: INTERVIEW FINDINGS - FACTORS ASSOCIATED WITH EFFECTIVE AND EFFICIENT CUSTOMER SENSEMAKING............................................................... 38
TABLE 9: INTERVIEW FINDINGS - FACTORS ASSOCIATED WITH SHARED IDENTITY AND GOALS ACROSS AND WITHIN DISTRIBUTED TEAMS......................................................... 39
TABLE 10: SUMMARY OF FIRM CAPABILITIES...................................................................................... 40
TABLE 11: SURVEY QUESTIONS........................................................................................................... 41
TABLE 12: AVERAGE AND SD SURVEY VALUES FOR THE 5 FIRMS WITH HIGHEST AND LOWEST GROWTH RATES.................................................. 42
TABLE 13: RESULTS FROM SPEARMAN'S CORRELATION TEST......................................................... 42
TABLE 14: IDENTIFIED ORGANIZATIONAL CAPABILITIES WITH ASSOCIATED KEY FACTORS............ 44
TABLE 15: MAPPING ORGANIZATIONAL CAPABILITIES TO PD ATTRIBUTES..................................... 48
TABLE 16: LIST OF PD PROCESS MODELS / FRAMEWORKS.............................................................. 52
TABLE 17: KEY CHARACTERISTICS OF MODELING PARADIGMS.................................................... 68
TABLE 18: PD ATTRIBUTES CAPTURED BY EACH MODELING PARADIGM........................................... 70
TABLE 19: PD ATTRIBUTES WITH CORRESPONDING MODELING VARIABLES................................ 83
TABLE 20: TASK DURATION ESTIMATES FOR A UV PROJECT............................................................. 108
TABLE 21: TASK RELATEDNESS MATRIX FOR UV PROJECT.............................................................. 109
TABLE 22: TASKS, THEIR TYPES AND CORRESPONDING DURATION ESTIMATES FOR MOBILE SOFT. DEV. PROJECT............................................................. 110
TABLE 23: DEVELOPER CHARACTERISTICS FOR MOBILE SOFT. DEV. PROJECT.............................. 111
TABLE 24: TASK TYPES AND DURATIONS FOR NETWORK IMPROVEMENT PROJECT........................ 113
TABLE 25: DEVELOPER CHARACTERISTICS FOR NETWORK IMPROVEMENT PROJECT.................. 113
TABLE 26: DEVELOPER CHARACTERISTICS FOR NEW FEATURE IMPLEMENTATION PROJECT........... 115
TABLE 27: TASK CHARACTERISTICS FOR NEW FEATURE IMPLEMENTATION PROJECT..................... 115
TABLE 28: DEVELOPER PARAMETERS FOR SCENARIO ANALYSIS...................................................... 128
TABLE 29: ROUTINES ASSOCIATED WITH ORGANIZATIONAL CAPABILITIES..................................... 163
TABLE 30: REWORK PROBABILITY MATRIX FOR UV PROJECT........................................................... 186
TABLE 31: REWORK IMPACT MATRIX FOR UV PROJECT................................................................. 186
TABLE 32: TASK RELATEDNESS MATRIX FOR MOBILE DEVELOPMENT PROJECT.............................. 187
TABLE 33: REWORK PROBABILITY MATRIX FOR MOBILE DEVELOPMENT PROJECT............................. 187
TABLE 34: REWORK IMPACT MATRIX FOR MOBILE DEVELOPMENT PROJECT...................................... 188
TABLE 35: BRIEF TASK DESCRIPTIONS FOR NETWORK IMPROVEMENT PROJECT............................. 188
TABLE 36: TASK RELATEDNESS MATRIX FOR NETWORK IMPROVEMENT PROJECT.......................... 189
TABLE 37: REWORK PROBABILITY MATRIX FOR NETWORK IMPROVEMENT PROJECT........................ 189
TABLE 38: REWORK IMPACT MATRIX FOR NETWORK IMPROVEMENT PROJECT.................................. 190
TABLE 39: OVERLAP MATRIX FOR NETWORK IMPROVEMENT PROJECT............................................ 190
TABLE 40: TASK RELATEDNESS MATRIX FOR NEW FEATURE IMPLEMENTATION PROJECT.................. 191
TABLE 41: REWORK PROBABILITY MATRIX FOR NEW FEATURE IMPLEMENTATION PROJECT (q = 1)........................... 191
TABLE 42: REWORK PROBABILITY MATRIX FOR NEW FEATURE IMPLEMENTATION PROJECT (q = 2)........................... 191
TABLE 43: REWORK IMPACT MATRIX FOR NEW FEATURE IMPLEMENTATION PROJECT (q = 1)............... 192
TABLE 44: REWORK IMPACT MATRIX FOR NEW FEATURE IMPLEMENTATION PROJECT (q = 2)............... 192

15
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABM</td>
<td>Agent-Based Modeling</td>
</tr>
<tr>
<td>API</td>
<td>Application Program Interface</td>
</tr>
<tr>
<td>CAGR</td>
<td>Compound Annual Growth rate</td>
</tr>
<tr>
<td>CDF</td>
<td>Cumulative Distribution Function</td>
</tr>
<tr>
<td>COUHES</td>
<td>Committee On the Use of Humans as Experimental Subjects</td>
</tr>
<tr>
<td>CPM</td>
<td>Critical Path Method</td>
</tr>
<tr>
<td>DES</td>
<td>Discrete Event Simulation</td>
</tr>
<tr>
<td>DR</td>
<td>Design Roadmap</td>
</tr>
<tr>
<td>DSM</td>
<td>Design Structure Matrix</td>
</tr>
<tr>
<td>GERT</td>
<td>Graphical Evaluation and Review Technique</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
<tr>
<td>HSC</td>
<td>Human Supervisory Control</td>
</tr>
<tr>
<td>IDEF</td>
<td>Integration DEFinition</td>
</tr>
<tr>
<td>IT</td>
<td>Information Technology</td>
</tr>
<tr>
<td>LE</td>
<td>Large Enterprise</td>
</tr>
<tr>
<td>LHS</td>
<td>Latin Hypercube Sampling</td>
</tr>
<tr>
<td>LIC</td>
<td>Low-Income Country</td>
</tr>
<tr>
<td>PD</td>
<td>Product Development</td>
</tr>
<tr>
<td>PDF</td>
<td>Probability Distribution Function</td>
</tr>
<tr>
<td>PDM</td>
<td>Precedence Diagram Method</td>
</tr>
<tr>
<td>PDSM</td>
<td>Product Development Simulation Model</td>
</tr>
<tr>
<td>PERT</td>
<td>Program Evaluation Review Technique</td>
</tr>
<tr>
<td>QA</td>
<td>Quality Analyst</td>
</tr>
<tr>
<td>RBV</td>
<td>Resource-Based View</td>
</tr>
<tr>
<td>SADT</td>
<td>Structured Analysis and Design Technique</td>
</tr>
<tr>
<td>Acronym</td>
<td>Definition</td>
</tr>
<tr>
<td>---------</td>
<td>------------</td>
</tr>
<tr>
<td>SD</td>
<td>Standard Deviation</td>
</tr>
<tr>
<td>SME</td>
<td>Small and Medium Enterprise</td>
</tr>
<tr>
<td>UAV</td>
<td>Unmanned Aircraft (or Aerial) Vehicle</td>
</tr>
<tr>
<td>UV</td>
<td>Unmanned Vehicle (of any type, e.g., land, sea, air)</td>
</tr>
<tr>
<td>WTM</td>
<td>Work Transformation Matrix</td>
</tr>
</tbody>
</table>
1. Introduction

1.1. Motivation
Cheaper, faster and better product development (PD) is crucial for success in today’s globalized and competitive markets. This has made the performance of PD projects an increasingly important factor to consider and improve for all types of projects. The market’s emphasis on getting faster and cheaper products has forced many enterprises to shift their PD practices from sequential to using predominantly concurrent and distributed practices. While on average this has reduced PD time, the dynamic complexity of the PD processes has significantly increased (Smith & Eppinger, 1997b; Wetherbe, 1995). This caused firms to try and develop a well-coordinated plan to organize their processes (Clark, 1991; Wheelwright, 1992). The increased complexity, however, has not been adequately accounted for by PD managers, who often utilize suboptimal decision heuristics (Kleinmuntz, 1993) to manage PD projects. Mental models utilized by PD managers and developers have not evolved adequately to allow for effective evaluation and management of projects (Ford & Sterman, 1998). This has contributed to poor management of many development projects, as noted by several researchers (e.g., Dertouzos, Lester, & Solow, 1989; Womack, Jones, & Roos, 2008). Eppinger et al. (1994) note that in some circumstances it may be impossible to even predict the impact of a single decision on the development process.

To improve PD processes, various researchers have developed models and frameworks based on the traditional Critical Path Method (CPM) (Shaffer, Ritter, & Meyer, 1965) and Program Evaluation Review Technique (PERT) (Roman, 1962), which statically describe the PD process. Other approaches have used the Design Structure Matrix (Steward, 1981) to capture more complex attributes of PD (e.g., iteration and information flow). More advanced simulation based modeling approaches have also been used. For example, Sterman (1998) utilized a system dynamic approach to model PD process. Other simulations paradigms, including discrete-event-simulation (DES) (e.g., Browning, 1999; Cho & Eppinger, 2005) and agent-based modeling have also been used to model PD processes to support decision making process.

While these models have the potential to significantly improve PD processes, there are important PD attributes that these models do not account for. Specifically, many simulation models do not explicitly account for resource constraints (Browning, 1999; Cho & Eppinger, 2001) and/or multiple geographically distributed PD teams. Furthermore, these models have been developed
and validated in the context of PD in large enterprises (LEs). Hence, the appropriateness of these models to small and medium enterprises (SMEs) merits exploration.

1.1.1. Addressing the need for small and medium enterprises
Product development process in LEs can be quite different from the PD process of SMEs (Ghobadian & Gallear, 1997). Hence, PD decision aid tools developed for LEs have to be adapted and tailored when managing PD processes of SMEs. Most commercially available tools have been developed for LEs, and academic research has focused on this context. This is understandable, since LEs have more resources to invest in the development of PD decision aid tools and also are financially more profitable to sell products to. Furthermore, LEs can hire outside consultants, if needed, to analyze and improve their PD processes.

Unlike LEs, SMEs generally have very limited resources to hire consultants and/or invest in the development of PD decision aid tools. Consequently, they resort to not using decision aid tools, using inappropriate tools or relying on ‘gut feel’. This situation gets exacerbated because many SMEs have been utilizing distributed PD as means of accelerating their PD timeline.

1.1.2. Accounting for distributed product development
Geographically distributed PD is no longer just a cost-cutting measure for many firms; rather it has become a stepping stone to increase competitive advantage by developing faster (and cheaper) products (Edoff & Srinivasan, 2013; Eppinger & Chitkara, 2006). Prevalence of distributed PD has been largely ignored in PD decision aid tools. Nonetheless, distributed PD can be notably different from co-located development. Specifically, coordination of developmental activities is of much greater importance when conducting distributed PD and needs to be taken into account. Several researchers have emphasized PD concurrency issues when the same project involves development activities across several locations (Tracy, 2012; Van Der Merwe, 1997). Hence, distributed PD adds an extra layer of difficulty for PD managers when analyzing and improving PD processes. Furthermore, the few tools that are available for distributed PD (e.g., Ghosh & Varghese, 2004) focus on tracking and managing tasks across various geographical locations; rather than analyzing and improving the process.
1.1.3. Creating a practical model / tool

An important factor for every PD model / decision aid tool is the practicality and easy of use. As Smith and Morrow (1999) note, most PD process models are not implemented in commercially usable software form and the ones that exist are “relatively crude and user-unfriendly.” Also, since past models are primarily developed in academic context, they have empirically justified modeling assumptions but often lack practical utility and relevance that firm PD managers are interested in. Moreover, past models generally have not been applied to guide decision making in real world situations, which negatively impacts practical value of these models.

Understanding how to improve PD processes of SMEs and being able to take into account distributed PD is crucial for developing a decision aid tool that is practical and valuable. Also, having the ability to use the tool to analyze and test various PD scenarios easily is of great value to PD managers. Furthermore, testing the tool with real world firms to prove its usefulness and improve its functionality is imperative for the practicality of the tool. Hence, the objective of this thesis was to develop and validate a practical PD process model / decision aid tool for analyzing and improving distributed PD processes of SMEs.

1.2. Research Questions

To address the above-mentioned thesis objective, the following research questions were posed:

- What are the important attributes and interactions that a PD process model / decision support tool needs to capture?
- Can the model be used to predict the impact of various PD attributes on project performance, such as project completion time and cost?
- What level of accuracy can be expected from the model? How does it compare to real world findings?
- Can a moderately trained individual use the tool without requiring the use of any supporting software?

1.3. Thesis Outline

To address these research questions, the remainder of this thesis is organized as follows:

- Chapter 2, Identifying Modeling Attributes, presents findings of past literature on PD of SMEs in low-income counties, as well as describes survey results administered to the
firms of interest. The chapter also identifies organizational capabilities that are important in organizing successful PD activities. Next, these capabilities are compared to literature findings regarding organizational capabilities of firms in developed countries. Lastly, organizational capabilities are utilized to elicit a list of PD attributes that need to be accounted for when building a PD model / decision aid tool.

- Chapter 3, Background: Modeling Product Development Processes, describes previously developed relevant PD models. Specifically, graph based, matrix based, analytical, and simulation models are presented, along with relevant attributes, advantages and disadvantages. Next, different simulation modeling paradigms are studied to assess their applicability in capturing PD attributes identified in the previous chapter. Existing PD process models do not capture all of the PD attributes of interest; hence, there is a need to develop a new model that fully captures these attributes.

- Chapter 4, Model Development, describes the modeling process that created the SimLink™ model, a DES based model of PD processes. The chapter presents relevant modeling constructs and describes the mapping of PD attributes to modeling constructs and variables. Next, DES based and PD specific model output metrics are presented.

- Chapter 5, Model Verification and Validation, presents the prototype implementation of the model for initial testing purposes, as well as more advanced implementation of the SimLink™ model in the form of desktop and mobile applications. Functionality of these applications is reviewed, along with the principles of model operations. The chapter then describes the verification and validation examples using a historical data set and several real world data sets. Lastly, model advantages are identified and sensitivity analysis is conducted.

- Chapter 6, Model Synthesis, demonstrates potential uses of the SimLink™ model. Data based on real world projects is used to analyze various scenarios and draw conclusions on ways to improve PD processes. The chapter also discusses generalizability of the model, along with its limitations.

- Chapter 7, Conclusions, summarizes important findings of this thesis research. Also, this chapter evaluates how well research objectives are met and research questions are answered. Finally, future work is suggested and both theoretical and practical contributions of this thesis are presented.
The research methodology used in this thesis is summarized in the graphical representation shown in Figure 1, which also outlines the relationships between the next chapters.

**Figure 1: Thesis research methodology.**
2. Identifying Modeling Attributes

2.1. Introduction

The economic success of most firms depends on their ability to identify the needs of customers and introduce new products over time (Dougherty & Hardy, 1996; Penrose, 1995). The goal of product development (PD) process is to create these products. PD is defined as the set of activities beginning with the perception of a market opportunity and ending in the production, sale, and delivery of a product (Ulrich & Eppinger, 2012). In today's environment of rapidly evolving customer preferences, speed and flexibility in developing new products is even more important (Takeuchi & Nonaka, 1986).

While majority of new product innovations have historically come from developed countries, Nielsen Company (2013) shows that less-developed countries (also referred to as low-income countries (LICs)) are gaining ground as new product innovators. Specifically, LICs contributed 31% of the world’s new product innovations in 2012, up by 6% from 2008 (Nielsen Company, 2013). Three key factors support the argument that the projected number of new product launches in developing countries will continue to grow faster than in the developed markets. First, the share of R&D conducted in LICs has been growing steadily and in 2012 India and China alone accounted for 20% of global R&D expenditure (Grueber & Studt, 2013). Second, four of the world’s most innovative 20 companies are now based in LICs. Furthermore, it is expected that half of the world’s top 500 companies will be headquartered in LICs by 2025 (Dobbs et al., 2013). Third, the consumption in emerging markets will, for the first time in history, be greater than consumption in developed countries (Atsmon, Child, Dobbs, & Narasimhan, 2012). It is projected that the next 35 years will add 2.5 billion people, 90% of them in less developed countries (Haub & Toshiko, 2012) who are generally more eager to try new products than Europeans and North Americans (Nielsen Company, 2013). Early examples demonstrate how new products have already been successfully exported from LICs to developed markets, as shown by Govindarajan and Trimble (2012).

While many large and small companies develop new and innovative products, this chapter focuses on product development processes of small and medium enterprises (SMEs) in LICs. The existing literature mainly emphasizes the role of large enterprises in PD (Adams Bigelow,
Research on how SMEs develop PD capabilities is limited (Mosey, 2005), despite the fact that SMEs represent the greatest share of total enterprises (e.g., 99% in Europe (European Commission, 2003)), and significant differences exist between large and small firm product development processes (Gibb, 2000). Christensen et al. (2002) argue that large firms innovate and develop products based on their already existing products by utilizing sophisticated technologies to improve performance of current products. SMEs, on the other hand, can compete by developing new-to-market products using new and often simpler technologies. Moreover, there are inherent differences between the management and structure of SMEs and Large Enterprises (LE) (e.g., Ghobadian & Gallear, 1997), which further supports the notion that the underlying PD processes are different. Welsh and White (1981) note, an SME cannot be viewed as a small version of a large company. Their viewpoint is supported by a recent study by Nicholas et al. (2011) who collected data from SMEs and LEs in Ireland and the UK and showed that SMEs and LEs do not consider the same practices to be ‘best practice.’

Studies conducted on SMEs that operate in LICs are even scarcer and most of the published research discusses possible ways of financially aiding SMEs (e.g., Beck, Demirgüç-Kunt, & Martínez Pería, 2008; Saeed, 2009) and do not focus on analyzing and improving PD processes. Hence, in this research, we consider firm-intrinsic success factors of SMEs that are either headquartered in a LIC or have a branch that operates in a LIC.

Furthermore, geographically distributed PD is increasingly becoming more popular (Granstrand, Håkanson, & Sjölander, 1993; Griffin, 1997). Various researchers have noted the importance of developing products in distributed fashion (e.g., Ghoshal & Bartlett, 1990) and with the emergence of reliable electronic-based communication media, firms are embracing distributed PD. Sproull and Kiesler (1992) note that information technology (IT) reduces the dependence on traditional face-to-face communications and creates “networked organization.” Because IT tools have become so widespread and affordable, many SMEs also utilize them to conduct distributed PD. Hence, in this research, a great effort was expanded to investigate SMEs that are involved in distributed PD.
2.2. Research Approach

An iterative, exploratory research design was used to identify PD success factors of SMEs who were either headquartered in Armenia or had a presence in Armenia. Our data sources included the current academic literature, interviews with key informants and quantitative project data. The literature review regarding SME product development and innovation in LICs was carried out to identify a preliminary set of organizational capabilities and routines. In parallel, we used seventeen key informant interviews carried out across ten Armenian firms to better understand their PD processes and identify the underlying organizational routines and emergent capabilities. Next, a survey was administered to the firms to collect more information and verify qualitative findings. Finally, PD data, such as the number of developers, teams and duration and complexity of various projects over time were collected to triangulate the qualitative findings. The data from all sources were synthesized to identify key organizational capabilities influencing PD. We had participant validation from the interviewed firms that we assessed to have been successful using review and feedback sessions.

2.2.1. Past literature findings

While most studies on success factors in PD have been conducted in the context of developed countries (Siu, 2005), a limited number of studies analyzed various factors impacting PD processes and innovativeness of firms in less-developed countries. For example, Hadjimanolis (1999) investigated innovation barriers for SMEs in Cyprus and found that there were some similarities with barriers in industrialized countries (e.g., lack of skilled labor and supply of finance). However, the study also found that there were many differences, such as inadequate technological infrastructure and government policies. Another study evaluated institutional factors affecting PD process of SMEs in China, Hong Kong, and Taiwan (Siu, 2005). The results suggest that various degrees of government intervention, different manufacturing systems and business approaches contribute to some differences in PD practices. While Korea is no longer a less developed country, a study published in 1993 by Kim et al. (1993) suggests that risk-taking propensity of small firm managers and their tolerance for ambiguity, along with professionalization of organizational structure and environmental heterogeneity are the most significant factors differentiating innovative from non-innovative small firms. Other studies have discussed the significant role that alliances (e.g., Adarkar, Adil, Ernst, & Vaish, 1997; Pisano,
Ireland, Hitt, & Webb, 2007), local manufacturing (e.g., Iyer, LaPlaca, & Sharma, 2006), and brands (e.g., Atsmon, Kuentz, & Seong, 2012) play in developing successful products and/or services for LICs. Nonetheless, studies discussing SME firm-intrinsic factors that significantly affect PD process in LICs is very limited. Moreover, there are no studies to our knowledge that investigate PD processes of SMEs conducting distributed PD in LICs. For this reason, interviews were conducted with a number of SMEs in Armenia to evaluate their PD processes.

2.2.2. Interviews and surveys
Semi-structured interviews with information technology (IT) firms in Armenia provided a window into actual PD process that companies use to create products (ones that the informants considered to be both innovative and non-innovative). Armenia’s entrepreneurial activity is ranked the highest in Caucasus by the World Bank (Kuriakose, 2013) and the growth in IT sector has been 27% CAGR between 1998 and 2009 (Enterprise Incubator Foundation, 2010). This is a result of many SMEs that have been founded in the country. In fact, the share of companies with fewer than 25 employees is above 70% in the IT sector (Enterprise Incubator Foundation, 2010). Hence, analyzing the PD process of SMEs in the IT sector in Armenia is an effective way to evaluate the factors influencing PD in LICs. Over six months, seventeen key informants from ten IT firms operating in Armenia were interviewed. Some firms had presence in multiple countries, while others conducted their development activities only in Armenia (but in many cases for an international client). Table 1 below provides information about the ten firms used to develop the routines and capabilities detailed in this chapter.

All interviews were conducted according to guidelines set by the committee on the use of humans as experimental subjects (COUHES) of MIT. The interviews had the same high-level structure (shown in Appendix A.1) to ensure that all interviewees answered the same set of questions. The interviews covered background information of the interviewee (e.g., area of specialization, experience level, current employment duration) and the company (e.g., product type, number, firm/management structure, and growth trends), PD process (e.g., description, practices, project management tools used, collaborations), and culture. The use of a semi-structured interview format provided the flexibility to accommodate any necessary deviations from the original script to engage the interviewee in a firm specific topic that emerged during the interview and was not covered by the interview guide. Each interview lasted between 50 minutes...
and two hours, with four of the informants being interviewed more than once to clarify key themes.

Also, a PD survey was administered to 16 firms (with 15 responses). The survey used a 5-point Likert scale to assess such PD related factors as flexibility of work hours, work-related update frequency, process modularity, and hiring criteria, among others. Appendix B.1 presents the survey that was distributed to the firms.

<table>
<thead>
<tr>
<th>Firm</th>
<th>Number of interviews</th>
<th>Brief description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>Developer of rich media advertisements for mobile platforms. Also, connects mobile advertisement content developers with publishers.</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Social media mobile app developer with emphasis on video sharing.</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>Mobile photo editor developer.</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Mobile games and game platform developer.</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>Developer of telecommunication systems platform of next generation.</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Mainly outsourcing firm developing mobile and web apps.</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>Web marketing firm offering graphic design, photography, and video production.</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>Developer of interactive marketing systems.</td>
</tr>
<tr>
<td>9</td>
<td>2</td>
<td>Branch of a large multinational firm that offers automated control systems and software testing solutions.</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>Mainly website development and support firm for the local market.</td>
</tr>
</tbody>
</table>

During the interviews, quantitative product development data was also collected. The next section summarizes this data for the ten firms described above.

2.2.3. Firm data

During interviews quantitative data was collected on the PD process of firms, as well as their observed growth rate in the last year and projected growth next year. Since half of the firms interviewed were involved in distributed PD, the number of locations where firms conduct PD activities was also collected. Furthermore, the number of developers (all employees, excluding administrative and support staff) was collected to estimate firm size. Finally, since some of the
firms were reluctant to provide financial information, the number of employees was used as a measure of firm success, as recommended by Witt (2004). Specifically, the percentage of increase/decrease of the number of developers was collected for the past year, along with the management’s projections for the next year. While the measure favors smaller firms (i.e., percentage increase/decrease is higher for smaller firms for the same absolute change in the number of employees), it provides sufficient context information for larger firms when the total number of developers is taken into consideration along with the growth rate. Once the growth rates for firms were calculated, they were coded in the following way (Table 2) using a five-point scale (-2 to 2).

Table 2: Firm growth/decline rate coding.

<table>
<thead>
<tr>
<th>Growth (+) or decline (-) rate of the number of developers (%)</th>
<th>Corresponding rating</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt; -20</td>
<td>-2</td>
</tr>
<tr>
<td>-20 to -10</td>
<td>-1</td>
</tr>
<tr>
<td>-10 to 10</td>
<td>0</td>
</tr>
<tr>
<td>10 to 20</td>
<td>1</td>
</tr>
<tr>
<td>&gt;20</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 3 below summarizes collected PD data.

Table 3: Firm descriptive data.

<table>
<thead>
<tr>
<th>Firm</th>
<th>Number of PD locations</th>
<th>Total number of developers</th>
<th>Growth rate last year</th>
<th>Projected growth rate next year</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>10-15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>10-15</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>50-60</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>40-50</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>10-15</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>10-15</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>40-50</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>5-10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Many</td>
<td>80-90</td>
<td>0</td>
<td>-1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>10-15</td>
<td>-1</td>
<td>-2</td>
</tr>
</tbody>
</table>

Besides descriptive firm data, PD project specific data was also collected from firms. More precisely, some firms provided data to help validate the simulation model. Chapter 5 presents several examples of corporate data collected from firms and used to conduct simulation analysis.
The data included a list of tasks corresponding to a specific project, developers working on that project, task durations, and task dependencies, among other parameters.

2.3. Identifying PD Capabilities

2.3.1. Introduction

The previous literature identifies three different perspectives for analyzing success factors (Witt, 2004): environmental factors, firm-intrinsic factors, and pre-and-post founding activities. As mentioned earlier, this thesis work focuses principally on firm-intrinsic factors to identify capabilities for successful PD. Since the firm-intrinsic view has to do with internal processes and insights (Darroch, 2005), resources of the firm are important for developing the right processes and insights. Firm resources include all assets, capabilities, organizational processes, firm attributes, information, knowledge, etc. controlled by the firm that enable the firm to conceive of and implement strategies that improve its efficiency and effectiveness (Barney, 1991). Resources are the cornerstones of competitive advantage of firms (Peteraf, 1993). The idea of studying firms as having a set of resources was first presented in the seminal work of Penrose (1995). Later this idea received more attention and the resource-based view of the firm (RBV) (Prahalad & Hamel, 1990; Wernerfelt, 1984) became a popular framework for analyzing competitiveness of firms. In recent years, RBV has been extended to dynamic markets (Teece, Pisano, & Shuen, 1997) to explain how and why some firms have competitive advantage during rapid and unpredictable change. Associated with firm’s resources, dynamic capabilities were introduced to represent “the firm’s processes that use resources – specifically the processes to integrate, reconfigure, gain, and release resources” (Teece et al., 1997). As Eisenhardt and Martin (2000) note, dynamic capabilities consist of identifiable and specific routines and they often exhibit commonalities across effective firms (also known as ‘best practice’). Various authors have used different naming conventions when referring to dynamic capabilities. For example, Henderson and Cockburn (1994) use the term ‘architectural competence,’ while Kogut and Zander (1992) used the term ‘combinative capabilities.’ In this paper, we will use the term ‘capabilities,’ similar to Amit and Shoemaker (1993). Generally, to be capable is to have a reliable capacity to bring that thing about as a result of intended action (Dosi, Nelson, & Winter, 2000). Many researchers use the terms capability and routine interchangeably. However, these terms have different meanings. Capability has a conscious, recognizable purpose expressed in terms of the significant
outcomes it is supposed to enable (e.g., develop innovative products) — as contrasted with the quasi-automatic character of operating routines, which are the building blocks of capabilities (Dosi et al., 2000). Nelson (1982) remarks that “what is regular and predictable about business behavior is plausibly subsumed under the heading ‘routine,’ especially if we understand that term to include the relatively constant dispositions and strategic heuristics that shape the approach of a firm to the non-routine problems it faces.”

2.3.1.1. Interview Data Analysis

An iterative analysis has been used to identify firm capabilities that drive successful PD. Tracy (2012, p. 184) mentions that “iterative analysis alternates between emergent readings of the data and an etic use of existing models, explanations, and theories.” Iteration is not a mechanical repetitive process, rather it is a reflexive process in which one visits and revisits data, generates new insights and progressively improves his/her understanding (Srivastava & Hopwood, 2009). To conduct this iterative analysis, a manual interview coding approach was adopted, in which the transcripts of the interviews were subjected to primary-cycle coding. In this phase, the data was examined and words or phrases were assigned to capture their essence. In our analysis, in vivo codes (Strauss, 1987) were used extensively to describe the data using the language and terms of participants themselves.

The primary coding analysis was conducted not just once, rather we used the constant comparative method (Charmaz, 2014) to compare data from various interviews and refine the terms that we are looking for when reading interview transcripts. After completing primary-cycle coding, we conducted secondary-cycle coding to synthesize and categorize findings from the previous cycles into interpretive, higher-level concepts. Tracy (2012, p. 194) mentions “secondary-cycle coding moves beyond first-level descriptive codes to analytic and interpretive second-level codes.” Second-level codes are often based on disciplinary concepts, hence previous literature findings are crucial when analyzing and coding interview data. During second-cycle coding, patterns and groupings of codes are also identified and the analysis switches from descriptive coding to more focused coding to try and identify the patterns in the data. Secondary-cycle repeats as many times as needed until theoretical saturation (Glaser & Strauss, 1967) is achieved, i.e., no new or relevant patterns emerge from analyzing new information.
During primary cycle coding thirteen factors related to successful PD are identified. Appendix A.2 presents these factors, along with quotes from interviews. During secondary cycle coding, these key factors were synthesized into a set of five capabilities, which the successful firms utilized, while the less successful firms did not. These five capabilities, along with associated key factors are shown in Table 4.

<table>
<thead>
<tr>
<th>Org. capabilities</th>
<th>Associated key factors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible PD</td>
<td>• Overlapping PD stages with short milestones</td>
</tr>
<tr>
<td></td>
<td>• Greater developer freedom</td>
</tr>
<tr>
<td></td>
<td>• Flexible workforce skills</td>
</tr>
<tr>
<td>Modular PD process</td>
<td>• Modular architecture</td>
</tr>
<tr>
<td></td>
<td>• Parallel task processing</td>
</tr>
<tr>
<td>Entrepreneurship focused</td>
<td>• Hiring criteria based on desire and ability to learn</td>
</tr>
<tr>
<td>human capital management</td>
<td>• Opportunities for professional growth</td>
</tr>
<tr>
<td></td>
<td>• Greater developer availability</td>
</tr>
<tr>
<td>Effective &amp; efficient customer</td>
<td>• Collect and integrate feedback often</td>
</tr>
<tr>
<td>sensemaking</td>
<td>• Utilize data mining</td>
</tr>
<tr>
<td></td>
<td>• Use focus groups</td>
</tr>
<tr>
<td>Shared identity and goals</td>
<td>• Uniting PD team around common goals</td>
</tr>
<tr>
<td>across and within teams</td>
<td>• ‘One team’ knowledge sharing culture across geographically</td>
</tr>
<tr>
<td></td>
<td>distributed locations</td>
</tr>
</tbody>
</table>

These capabilities are further discussed in the next sections.

2.3.2. Flexible product development

Many models of PD are based on the idea that design and development activities are best divided into a number of sequential project 'stages' separated by milestones called 'gates' (R. G. Cooper, 1990; Ulrich & Eppinger, 2012). However, the stage-gate process has been shown not to be effective in dynamic environments (Bhattacharya, Krishnan, & Mahajan, 1998; Iansiti, 1998). Recently, more flexible models of product development have been explored (e.g., Krishnan, Eppinger, & Whitney, 1997) and these models have been widely advocated to be used in software development (e.g., Boehm, 1988).

Firms 1 to 3 and 6 had flexible product development approaches, such that various stages of PD process overlap, thereby allowing feedback from later stages of PD to flow easily and still affect
early stages of development. To further facilitate feedback integration mechanism, developers in these firms had milestones at a minimum every two days. This not only allowed even little feedback to be used to guide future development but has also been shown to improve developer performance and reduce development time due to ‘time-boxing’ (Massey, Montoya-Weiss, & Hung, 2003).

Some teams took the concept of flexibility and applied it to their development teams as well. They empowered the team with greater flexibility in terms of the hours, location, and, to some degree, even the type of work they wanted to do. In fact, firms 1 and 2 did not have fixed hours for any of the developers, outside of the required team meeting times. The freedom to choose work hours and location preference helped accelerate the development process because it allowed the management team to dynamically adjust the workload and hours based on the development schedule. For example, during extra high workload periods, the management team of firm 2 utilized a part-time three-person team. In case of firm 1, developers occasionally were asked to work from home during weekends.

Flexibility in PD was also manifested by developers’ freedom to choose and experiment how to implement tasks assigned to them, i.e., developers were involved in doing creative work, rather than implementing what had already been decided for them.

Compared to firms 1 and 2, firm 10 was on the opposite spectrum of flexibility. In fact, the PD process was not flexible, which resulted in significant amount of rework near the end of projects. Also, work hours were strictly fixed to be 9am through 6pm, which created a negative attitude towards the management and developers often could not wait to leave the firm’s office at 6pm.

Table 5 summarizes key factors associated with the flexible PD capability, along with interview findings. The ‘checkmark’ (✓) indicates that during interviews the corresponding key factor was either explicitly mentioned or implied. While the ‘x’ mark (✗) implies that an approach opposite to the identified factor was mentioned or implied during interviews.
2.3.3. Modular PD

The concept of modularity is closely tied with flexibility (Sanchez, 1993). According to Liang and Huang (2002), modular product development is one in which the input and output relationships between components in a product have been fully standardized and specified. The need to respond to new information in the later stages of development requires a product architecture that can incorporate this new information without requiring time-consuming and expensive changes to the system (MacCormack, Verganti, & Iansiti, 2001). Besides accommodating future uncertainty, modularization also makes complexity manageable and enables parallel work (Baldwin & Clark, 2006). Furthermore, modularity is partially responsible for allowing effective subcontracting, since interface specifications for various components are standardized (Eggen, 2003). Hence, it should be expected that all three successful firms that had multiple development sites (e.g., firms 1-3) paid significant attention to building modular products. In the case of firm 2, once the concept of the app was approved, the architecture was refactored with modularity in mind. Later in the development stages, the modular design remained at the core of the app. One way firm 2 developers implemented modular architecture design was to load most user interface related information from a back-end server every time the app was launched. Specifically, the placement of interface buttons, colors, and functionality was not hard-coded, which allowed necessary changes to be implemented very quickly without the need to change the code of the app. In fact, the existing app architecture makes it straightforward to expand the operating platform of the app to Android and Windows systems.

Table 6 below summarizes interview findings regarding modular PD capability.
Table 6: Interview findings - factors associated with modular PD.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular architecture</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Parallel task processing</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.3.4. Entrepreneurship focused human capital management

Given that most software development work in a LIC is in the form of IT outsourcing with very limited potential to work on developing innovative products, the importance of professional growth opportunities cannot be overemphasized. The management of firms 1 and 2 allowed developers to work on challenging problems that did not directly add immediate value to the project. For instance, a firm 2 developer worked full-time on the back-end analytics for more than a month. This was a challenging feature to implement that was not an obvious value add (during initial phases of app development), however, it became a key component to enable product improvements. This commitment to let individual developers own and work on large pieces of the project further motivated developers and contributed to their professional growth.

Also, successful firms have created an environment where entrepreneurship is encouraged. In fact, the founder of firm 2 was determined to shift developer culture from thinking of their jobs as “nine-to-six” undertaking without much stimulating work, to a flexible work arrangement with plenty of opportunities to grow not only their professional skills, but also their leadership abilities. Furthermore, the founder of firm 4 was willing to help his employees start their own startups, even on the expense of losing them as employees.

Last but not least, firms 1 to 4 hired employees based on their ability to learn and desire to work to develop innovative products. This should be contrasted to hiring criteria of the less successful firms, which were mainly based on the past work experience and on the amount of starting salary.

Table 7 below summarizes interview findings regarding entrepreneurship focused human capital management PD capability.
Table 7: Interview findings - factors associated with entrepreneurship focused human capital management.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hiring criteria based on desire and ability to learn</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Opportunities for professional growth</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greater developer availability</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

2.3.5. Effective & efficient customer sensemaking

The importance of user/customer feedback in the PD process varies based on the nature and purpose of the product being developed, as well as the stage of the PD process. Nonetheless, successful firms tend to actively collect feedback, while less successful firms either do not collect feedback at all or wait until feedback (usually negative) is provided.

Due to the nature of the main products of firms 2, 3, and 4 (i.e., user-centric), it was crucial for these firms to be able to collect early and reliable feedback from users. Von Hippel (1986) showed the importance of linking the target market to product development process as early as possible. All of these firms engaged potential customers very early in the development process, while firms 2, 4, 5 also utilized back-end analytics to supplement the data collected from users.

The type and the extent of user engagement varied based on the development phase. For example, firm 2 initially conducted only field explorations to validate and evaluate the mobile application idea. Next, limited focus groups and informal interviews were conducted to refine the concept and identify useful features. Finally, extensive focus groups and product trials were performed to further refine the application, analyze usage patterns, and explore opportunities for commercial multi-city launch. The management of firm 2 also cultivated a community of users to evaluate the product in depth, which helped the team collect rapid feedback (Iansiti, 1998; Thomke, 1998).

Other successful firms have hired data analysts (e.g., firm 1) to draw conclusions from the vast amount of customer data. Firm 4 utilized another interesting strategy of collecting user feedback. More specifically, the management asked expats from different countries living in Armenia to provide feedback on the firm’s products to gauge differences on how different cultures perceived the same product. They also launched products in test markets (e.g., New Zealand, Australia)
before launching the product in the main markets (e.g., USA).

It is worth mentioning, once again, that during the initial product development stages, it was not easy to see the value that a back-end analytics engine would add to the development of a mobile application. Given that back-end analytics does not directly add value to near term user interaction, funders and engineers are often unwilling to spend resources on developing a well-structured back-end analytics engine. However, the management of firms 2 and 4 were determined to build a powerful analytics engine to support further PD. Despite the fact that both of these firms collected user feedback by many means, back-end analytics had become the principal tool for product improvement.

Table 8 shows interview findings of key factors for each of the ten firms.

<table>
<thead>
<tr>
<th>Table 8: Interview findings - factors associated with effective and efficient customer sensemaking.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collect and integrate feedback often</td>
</tr>
<tr>
<td>Utilize data mining</td>
</tr>
<tr>
<td>Use focus groups</td>
</tr>
</tbody>
</table>

2.3.6. Shared identity and goals across and within geographical locations

The driving force of any idea is the team behind it. Effective coordination of tasks between conventional team members is important but it becomes essential in globally distributed teams because they experience conflict and knowledge-transfer problems more often than conventional teams (Cramton, 2001a; Mannix, Griffith, & Neale, 2002). Past research has shown that the underlying dynamics of distributed teams differ from those of co-located teams (Gibson & Cohen, 2003; Hinds & Mortensen, 2005). Hence it should not be surprising that all successful firms with two geographical PD locations had taken deliberate steps to create unifying culture and promote shared identity across geographical locations.

All founders (or cofounders) of firms 1, 2, and 3 interviewed stated that developers and managers understand that if the product fails, then everyone fails. This common understanding unites everyone towards achieving a common goal, which has been shown to be effective in fostering team coordination (Pettigrew, 1998; Sidhu & Volberda, 2011). The management of
firms 1 and 2 often understaffed the development teams to further motivate the developers to reach out and collaborate to accomplish their tasks. This encouraged developers to share ideas with each other and learn something new. This effectively encouraged horizontal flow of information and mutual knowledge building between team members working on interdependent tasks, which research indicates enhances task execution (Galbraith, 1995; Krauss & Fussell, 1990; Thompson, 1967). Furthermore, understaffing also creates what Rousseau (1998) calls “situated [shared] identification”, which makes team members cognitively expand to the level of the team.

Another way shared identity was promoted in firms 1 and 2 was to encourage “direct” communication among all team members both in different geographical locations. This encouraged more horizontal collaboration and resulted in less vertical information transfer, which often causes information loss (Allen, 1977).

Also, direct access to users was mentioned as another characteristic that unifies the teams and encourages developers. This three-way interaction (e.g., between the team in Armenia, in the US, and clients) has been shown to further facilitate task coordination (Sidhu & Volberda, 2011).

Table 9 shows interview findings of key factors for each of the ten firms.

Table 9: Interview findings - factors associated with shared identity and goals across and within distributed teams.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uniting PD team around common goals</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>‘One team’ knowledge sharing culture (across geographically distributed locations)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>↑</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

2.4. Synthesis

2.4.1. Summary of firm capabilities

To compare the firms studied in this paper, the identified capabilities are evaluated for each of the ten firms. Table 10 summarizes the results. More specifically, the check mark (✓) implies that at least one of the factors corresponding to the capability is present; the square (■) implies that the correspondent capability is not being utilized but it is also not being suppressed; finally, the x mark (×) implies that at least one of the factors corresponding to the capability is being
suppressed. In short, Table 10 summarizes the information presented in Table 5 to 9. As can be seen from the table, the top three successful firms actively utilize all of the identified capabilities, while the least successful firms mostly do not utilize these capabilities. Moreover, as firms become less successful (i.e., the past and future growth rates of firms decline), the number of capabilities that are not present and/or are not utilized increases. This shows that there is a correlation between how successful the firms are and how many of the identified capabilities are being utilized/present (discussed further in section 2.4.2).

Table 10: Summary of firm capabilities.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible PD</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Modular PD</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Entrepreneurship focused human capital management</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Effective &amp; efficient customer sensemaking</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Shared identity and goals across and within geographical locations</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

To validate the findings more rigorously, a survey has been developed and distributed to the firms evaluated in this study. The next section presents the results of the survey. Furthermore, routines associated with the capabilities have been identified from interviews and are presented in Appendix A.3. These routines can be used by SMEs to help build organizational capabilities for successful PD.

2.4.2. Survey results

The survey results were collected from 16 firms. The data collected from one of the firms was not used in the analysis because the answers to all of the questions were the same, except to the last question. Table 11 presents the questions used in the survey. Questions 1 to 4 assess firm’s flexibility and questions 5 to 8 deal with modularity. Furthermore, questions 9 and 10 relate to entrepreneurship focused human capital management, questions 11 to 14 relate to the customer feedback collection mechanism and importance, while 15 to 17 relate to the firm’s culture. Appendix B.2 presents firms’ responses to all of the questions, along with the survey distributed to the firms.
Table 11: Survey questions.

<table>
<thead>
<tr>
<th></th>
<th>Question</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>How flexible are work hours of developers?</td>
</tr>
<tr>
<td>2</td>
<td>What is duration of average task/ticket assigned to developers?</td>
</tr>
<tr>
<td>3</td>
<td>How often do developers provide updates about their progress to the product/project manager (or to their team)? (Either formally or informally)</td>
</tr>
<tr>
<td>4</td>
<td>How often do developers work outside of their main area of expertise?</td>
</tr>
<tr>
<td>5</td>
<td>How similar is your product development process from product to product?</td>
</tr>
<tr>
<td>6</td>
<td>On average, how modular is each software/product developed by your firm? (More specifically, does each individual software/product component represent a complete subunit with a well-specified interface?)</td>
</tr>
<tr>
<td>7</td>
<td>In general, how modular is your product development process?</td>
</tr>
<tr>
<td>8</td>
<td>What is the maturity of the product architecture before new product development starts?</td>
</tr>
<tr>
<td>9</td>
<td>What are the hiring criteria for new employees? Specifically, what are the most important attributes for hiring a new person?</td>
</tr>
<tr>
<td>10</td>
<td>How much attention is paid to helping employees develop professionally?</td>
</tr>
<tr>
<td>11</td>
<td>How important is customer feedback to developing new products?</td>
</tr>
<tr>
<td>12</td>
<td>How often do you collect customer feedback?</td>
</tr>
<tr>
<td>13</td>
<td>How often rework is created due to customer feedback?</td>
</tr>
<tr>
<td>14</td>
<td>How is customer feedback collected? (Choose all that apply)</td>
</tr>
<tr>
<td>15</td>
<td>How important is shared identity across geographical locations? (If applicable)</td>
</tr>
<tr>
<td>16</td>
<td>How important is it for your firm’s employees to have common goals with respect to products they work on?</td>
</tr>
<tr>
<td>17</td>
<td>How important is it to have unifying culture?</td>
</tr>
</tbody>
</table>

The average and standard deviation values for each group of questions was calculated for the five firms with the highest combined growth rates (i.e., growth rate from last year and projected rate for the current year) and for the five firms with the lowest growth rate. Table 12 shows the results of this comparison.

The largest difference is observed for the modular product design and effective and efficient customer sensemaking. This is most likely due to more long term planning capabilities that the firms with higher growth rates have. Specifically, these firms pay more attention to the lifecycle development of their products, hence they need to better organize product’s architecture and modular approach is preferred by many PD managers. Also, for products to be successful in the long run, potential customers’ or clients’ feedback is crucial in the development process. This is the primary reason that growing firms pay more attention to efficiently and effectively incorporating customer feedback in the PD process.
Table 12: Average and SD survey values for the 5 firms with highest and lowest growth rates.

<table>
<thead>
<tr>
<th></th>
<th>Highest growth rate</th>
<th>Lowest growth rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>SD</td>
</tr>
<tr>
<td>Flexible PD</td>
<td>3.3</td>
<td>0.8</td>
</tr>
<tr>
<td>Modular product design</td>
<td>4.2</td>
<td>0.1</td>
</tr>
<tr>
<td>Entrepreneurship focused human capital management</td>
<td>4.0</td>
<td>1.0</td>
</tr>
<tr>
<td>Effective and efficient customer sensemaking</td>
<td>4.3</td>
<td>0.3</td>
</tr>
<tr>
<td>Shared identity and goals across and within geographical locations</td>
<td>4.25</td>
<td>0.31</td>
</tr>
</tbody>
</table>

To understand whether there is any statistical difference between firm growth rates and capabilities, the results of the survey were used to run statistical tests. Because the data collected did not meet normality assumptions, a non-parametric Spearman’s correlation test was used. This test measures the direction and strength of association between capabilities and firm growth rates. Table 13 shows Spearman’s rank correlation coefficient ($\rho$) corresponding to each capability. Statistically significant ($p = 0.05$) correlations are shown with a ‘*’ sign.

Table 13: Results from Spearman’s correlation test.

<table>
<thead>
<tr>
<th>Capability</th>
<th>Spearman’s rank correlation coefficient ($\rho$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible PD</td>
<td>0.53</td>
</tr>
<tr>
<td>Modular product design</td>
<td>0.84*</td>
</tr>
<tr>
<td>Entrepreneurship focused human capital management</td>
<td>0.66*</td>
</tr>
<tr>
<td>Effective and efficient customer sensemaking</td>
<td>0.71*</td>
</tr>
<tr>
<td>Shared identity and goals across and within geographical locations</td>
<td>0.73*</td>
</tr>
</tbody>
</table>

The results indicate that except flexible PD capability, all others have statistically significant correlations with firms’ growth rates. This is important, since it validates earlier findings that these capabilities play a crucial role in managing the development of new products. One potential reason that flexible PD capability is not significantly correlated with firm growth rate is the fact that the questions in the survey covered two distinct aspects of flexibility – flexibility in terms of developers’ hours and skills and flexibility in terms of task splitting and providing
updates. Since some firms were flexible in one aspect but not the other, this caused the survey results regarding flexible PD to be inconsistent, which impacted the significance level of Spearman’s correlation test.

The survey results have significant practical implications, since they show that firms that can build the five capabilities mentioned above and utilize them will most likely also succeed in building effective and efficient PD processes. It should also be noted that these capabilities in concert are much more effective than in isolation. This can also be observed from the interview and survey results – successful firms usually have all (or most) of the capabilities.

The next section analyzes these capabilities in the context of SMEs operating in developed countries.

2.4.3. Comparing capabilities with successful firms operating in developed countries

Once organizational capabilities were identified, the key factors associated with capabilities were compared with previous literature findings. The results showed that findings are not unique to the firms operating in LICs, rather they are more general and apply to firms that are based in developed countries. This finding should not be surprising after all – successful firms studied in this thesis developed products not just for the local markets but also for global clientele and many of these firms also operated in developed countries. Hence, it is logical that the identified capabilities can be generalized to SMEs operating in industrialized countries. Table 14 shows capabilities and associated key factors, along with example references from past literature.

Because of the similarities in past literature findings and current research results, analysis conducted later in this thesis can be generalized to SMEs operating in developed countries. This also includes the PD process model that is presented in Chapter 4.

2.5. Affected PD Attributes

To identify which PD attributes each of the above-mentioned capabilities impacts in a significant way, an iterative analysis, described in section 2.3.1.1, was conducted. First, a core list of PD attributes from past PD process models and literature was identified (e.g., Browning, 1999; Cho & Eppinger, 2001; Ford & Sterman, 1998). These attributes are:
• *Task structure* includes information regarding task types, breakdowns, and their durations.

• *Task relatedness* refers to the information flow between tasks, such as task dependencies and parallel and sequential task processing.

• *Rework* refers to the probability that a task will need to be attended more than once and the impact that this will have on the duration of the task.

• *Learning* refers to the reduction of task duration from working on the same task (or task type) more than once.

To find additional PD attributes that are impacted, the interview transcripts were studied. Specifically, the key factors associated with organizational capabilities (Table 4) were analyzed to identify PD attributes they impact. In the next sections, PD attributes corresponding to each capability are acknowledged and discussed.

<table>
<thead>
<tr>
<th><strong>Org. capabilities</strong></th>
<th><strong>Associated key factors</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible PD</td>
<td>• Overlapping PD stages with short milestones (MacCormack et al., 2001)</td>
</tr>
<tr>
<td></td>
<td>• Greater developer freedom (Kelliher &amp; Anderson, 2010)</td>
</tr>
<tr>
<td></td>
<td>• Flexible workforce skills (Edwards, 1993)</td>
</tr>
<tr>
<td>Modular PD process</td>
<td>• Modular architecture (Dahmus, Gonzalez-Zugasti, &amp; Otto, 2001)</td>
</tr>
<tr>
<td></td>
<td>• Parallel task processing (Baldwin &amp; Clark, 2006)</td>
</tr>
<tr>
<td>Entrepreneurship focused human capital</td>
<td>• Hiring criteria based on desire and ability to learn (Hayton, 2003)</td>
</tr>
<tr>
<td>management</td>
<td>• Greater developer availability (Ruth Eikhof, Warhurst, Haunschild, Bergman, &amp; Gardiner, 2007)</td>
</tr>
<tr>
<td></td>
<td>• Opportunities for professional growth (Lau &amp; Bruce, 1998)</td>
</tr>
<tr>
<td>Effective &amp; efficient customer</td>
<td>• Collect and integrate feedback often (Von Hippel, 1986)</td>
</tr>
<tr>
<td>sensemaking</td>
<td>• Utilize data mining (Berry &amp; Linoff, 1999)</td>
</tr>
<tr>
<td></td>
<td>• Use focus groups (Thomke, 1998)</td>
</tr>
<tr>
<td>Shared identity and goals across and</td>
<td>• Unite team around common goals (Sidhu &amp; Volberda, 2011)</td>
</tr>
<tr>
<td>within teams</td>
<td>• ‘One team’ knowledge sharing culture across geographically distributed locations (Kotlarsky &amp; Oshri, 2005)</td>
</tr>
</tbody>
</table>
2.5.1. Attributes impacted by flexible PD

This section identifies the main PD attributes that flexible development impacts, i.e., these attributes can be impacted significantly when a firm uses a flexible vs. non-flexible PD approach. The flexible PD capability has been described in section 2.3.2 and has the following main properties: (a) overlapping PD stages and short milestones, (b) greater developer freedom, and (c) flexible workforce skills.

The PD attributes needed to account for these properties are:

- *Task structure* to set short task durations and breakdowns.
- *Task relatedness* to account for overlapping of tasks and parallel/sequential processing.

Besides these attributes that are identified in the previous literature, additional new attributes are impacted, which are presented next:

- *Team structure* to be able to change the number of developers in the PD team to match the work demand.
- *Developer flexibility*, which accounts for varying developer hours and developer skill levels and competencies.
- *Team coordination* to account for passing of tasks between geographically distributed PD teams.

It should be noted that there are other PD attributes that flexible development techniques may impact implicitly, which are not mentioned above. For example, *rework* attribute will be impacted due to changing task structure. However, this is not a primary effect of flexible PD and for that reason has not been noted above.

2.5.2. Attributes impacted by modular PD

Modular PD allows for division of tasks in a way that promotes parallel processing and allows for more flexible, modular process architecture. Hence, it is natural that many PD attributes are impacted by modular PD process architecture. These attributes are presented below.
• *Task structure* can be greatly impacted when using modular process architecture. Specifically, process architecture affects the structure / arrangement of tasks in a process, hence, task structure.

• *Team structure* also can change depending on the architecture of the development process. Particularly, for a specific PD process architecture, certain developers may be required for a certain period of time, which is captured in the *team structure* PD attribute.

• *Task relatedness* is another attribute that is significantly impacted. To architect an efficient and predictable PD process implies arranging tasks in a way that improves information flow between tasks, such that task processing delays and rework is minimized. This means that the task relatedness attribute is crucial in terms of capturing the effects of modular PD process.

• *Team coordination* is another attribute that will be impacted by the choice of PD process architecture, if the PD involves multiple teams. For instance, if a PD project involves splitting tasks between teams, then the change of architecture will impact the splitting and passing of tasks between members of different teams.

### 2.5.3. Attributes impacted by entrepreneurship focused human capital management

Entrepreneurship focused human capital management emphasizes the need for developers grow professionally, even at the expense of the firm (in short term) by spending resources on developers to acquire new skills and motivate them. This involves encouraging creativity, trusting developers with challenging problems and hiring based on desire and ability to learn rather than on past experience alone.

The two main PD attributes that are needed to capture the effects of entrepreneurship focused human capital management, are *developer flexibility* and *learning/performance attributes*.

• *Developer flexibility* is impacted to the extent that developers are encouraged to work more flexible hours, as well as broaden their skillset and work on tasks that are outside of their area of expertise.

• *Learning* attribute captures the reduction in task completion time when the task is repeated. Hiring based on the ability and desire to learn and grow can significantly impact this attribute.

46
• Moreover, because employees can have a wide range of performance characteristics, it is important to capture these differences. The performance PD attribute can change based on employee hiring criteria. For example, employees with several years of experience can have a higher performance factor but lower learning abilities, while someone with very little experience can learn very quickly but have a lower initial performance factor.

2.5.4. Attributes impacted by effective and efficient customer sensemaking

Effective and efficient customer sensemaking encourages involving customers early and often in the PD process. Also, various techniques are suggested to collect feedback quickly and efficiently. The main PD attributes that can be impacted are task structure and rework.

• Task structure is impacted because collecting feedback efficiently and often requires structuring the tasks so that feedback collection tasks can be incorporated into the PD process more frequently.
• Rework is often required after analyzing customer feedback and incorporating new information into the PD process.

2.5.5. Attributes impacted by shared identity and goals

The last capability aims to unite geographically distributed teams around a common goal and develop shared identity, encourage collaboration among team members and eliminate communication barriers. The PD attributes that are primarily impacted by this capability are presented next.

Task relatedness may be altered to account for parallel and sequential task processing between different distributed teams based on the level of cooperation between the teams.

Team coordination is also impacted and can vary from difficult to coordinate (for firms that do not develop shared identity and/or goals) or easy to manage for firms with better ability to unite the distributed teams.

The next section summarizes the impacted PD attributes corresponding to each capability identified in section 2.3.
2.5.6. Summary of PD attribute mapping

To the list of core PD attributes (i.e., task structure, task relatedness, rework, and learning) identified in past modeling literature, other attributes were added based on the analysis of organization capabilities identified in this chapter. Table 15 maps the organizational capabilities to the PD attributes that each capability impacts in a meaningful way.

Table 15: Mapping organizational capabilities to PD attributes.

<table>
<thead>
<tr>
<th>Flexible PD</th>
<th>Modular PD process</th>
<th>Entr. focused human capital mngmt</th>
<th>Effective &amp; efficient customer sensemaking</th>
<th>Shared identity and goals</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task structure</td>
<td>■</td>
<td>■</td>
<td>■</td>
<td></td>
</tr>
<tr>
<td>Task relatedness</td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Developer flexibility</td>
<td>■</td>
<td></td>
<td>■</td>
<td>■</td>
</tr>
<tr>
<td>Rework</td>
<td></td>
<td></td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Team structure</td>
<td>■</td>
<td>■</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Team coordination</td>
<td>■</td>
<td>■</td>
<td></td>
<td>■</td>
</tr>
<tr>
<td>Learning / Performance</td>
<td></td>
<td>■</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The table shows that besides the core PD attributes identified earlier, there are other PD attributes that need to be captured in a PD process model to account for the impact of organizational capabilities. These attributes primarily emerge from the fact that in SMEs individual developers (and teams of developers) play a significant role in the PD process compared to large firms, where the concept of an ‘average’ developer is sufficient to characterize PD processes. These newly identified PD attributes are described below, along with the core PD attributes (for completeness).

- *Task structure* includes information regarding task types, breakdowns, and their durations.
- *Task relatedness* refers to the information flow between tasks, such as task dependencies and parallel and sequential task processing.
- *Developer flexibility* accounts for the range of developer skillset and work hours.
• **Rework** refers to the probability that a task will need to be attended more than once and the impact that this will have on the duration of the task.

• **Team structure** captures parameters of different teams working on the same project. Specifically, team structure includes information about the number of developers assigned to a each team, along with corresponding skillsets and performance and learning factors.

• **Team coordination** refers to the extra coordination effort that needs to be expanded when passing tasks from one team to another.

• **Learning** refers to the reduction of task duration from working on the same task (or task type) more than once.

### 2.6. Summary

To identify attributes necessary for developing a PD process model, a qualitative was conducted. First, past literature was surveyed to extract relevant information regarding PD in SMEs. Next, interviews were conducted with SMEs (mainly in IT industry) to get in depth information regarding their PD processes and success factors. After primary cycle coding, thirteen factors were identified from interviews to have a significant impact on the success of PD processes. The secondary cycle coding helped group these factors into a set of five organizational capabilities that successful firms had, while the less successful firms did not have these capabilities. Also, a survey was developed and distributed to the firms to quantitatively assess the correlation between the five organizational capabilities and firm growth rates. The results showed that four out of the five organizational capabilities strongly correlate with firm growth rates. In the next step of the analysis, key factors associated with the capabilities were utilized to extract a list of PD attributes that the five capabilities impact. This list consisted of four attributes identified in previous literature and of three new attributes.

In the next section, various PD process models and frameworks are analyzed and their applicability in modeling distributed PD in SMEs is considered. Also, the ability of different simulation modeling paradigms to take into account PD attributes identified in this chapter is discussed.
3. Background: Modeling Product Development Processes

Various PD models have been developed and utilized successfully in the past. While models vary from representing the PD process with simplified visual diagrams to complex analytical and simulation based methods, ultimately, the goal of process modeling is to help define, size, and plan activities and deliverables and improve our general understanding of PD processes (Lévárdy & Browning, 2009; Whitney, 1990). A model is an abstract representation of reality that is built, verified, analyzed and manipulated to increase understanding of that reality (Browning, Fricke, & Negele, 2006). It has also been mentioned that “all models are wrong, while some are useful” (Launer & Wilkinson, 1979). Models are wrong because they cannot represent reality completely and accurately (i.e., models are abstractions). However, useful models provide insights that otherwise may be hard to get in real life. Some useful models have higher theoretical power, while others are more practical. However, there are important interactions between the practical value of the model and its theoretical power (Brady et al., 1997). Researchers and practitioners decide based on their needs and resources which model is appropriate for each scenario, given the model’s advantages and disadvantages. Several surveys of PD modeling literature were conducted in the past. Although these surveys have different foci, they review predominantly the same models; however, group them differently. Smith and Morrow provide an excellent review of engineering models of PD processes. A more recent review by Browning and Ramasesh (2007) also focuses on engineering models and covers slightly larger set of previously developed models for managerial purposes. Browning et al. (2006) and Amigo et al. (2013) reviewed various modeling frameworks according to their key variables and modeling characteristics. Elmaghraby (1995) focuses on activity networks using computer simulations for generic project management. Other researchers, such as Finger & Dixon (1989) and Kusiak (1999) provide a literature review of PD models for engineering purposes. Jones (1992) provides another outstanding review of the models that were used in 1960s and earlier. Since, most of the models developed later either incorporate earlier models or present new and innovative methods, models covered by Jones (1992) review article are not presented in this thesis. It is worth noting that several researchers have also conducted reviews of PD without emphasizing process modeling. For example, Shane and Ulrich (2004), Gerwin and Barrowman (2002) provide an excellent (and generic) review of PD literature.
This chapter presents a review of main PD process modeling frameworks, which are grouped based on their representation of the PD process. Nonetheless, at the core of all the frameworks is the assumption that PD is composed of a number of tasks with an underlying structure. This assumption has its roots in studies conducted more than half a century ago (e.g., Alexander, 1964; Marples, 1961). As described in the previous chapter, time is one of the main parameters that interests project managers, hence, the models presented below take into account project scheduling and time, some more accurately (and explicitly) than others. Table 16 below summarizes PD process related frameworks that are further discussed in this chapter.

Table 16: List of PD process models / frameworks.

<table>
<thead>
<tr>
<th>Graph based</th>
<th>Matrix based</th>
</tr>
</thead>
<tbody>
<tr>
<td>o Directed graph</td>
<td>o Design structure matrix / activity-activity incidence matrix</td>
</tr>
<tr>
<td>o CPM/PERT</td>
<td>o Work transformation matrix</td>
</tr>
<tr>
<td>o GERT/Q-GERT</td>
<td></td>
</tr>
<tr>
<td>o SADT/IDEF</td>
<td></td>
</tr>
<tr>
<td>o Petri net</td>
<td></td>
</tr>
<tr>
<td>o Signal flow graph</td>
<td></td>
</tr>
<tr>
<td>o Design roadmap</td>
<td></td>
</tr>
<tr>
<td>o Precedence diagram method</td>
<td></td>
</tr>
<tr>
<td>o Gantt chart</td>
<td></td>
</tr>
<tr>
<td>o Mathematical programming</td>
<td>o Design structure matrix</td>
</tr>
<tr>
<td>o Queuing theory</td>
<td>o Work transformation matrix</td>
</tr>
</tbody>
</table>

3.1. Graph Based

3.1.1. Directed graph

Directed graphs (or digraphs) provide a very intuitive way of representing processes (Kusiak & Wang, 1993). Digraphs consist of nodes, which represent tasks and edges depicting information transfer between tasks. Directed graphs are useful when modeling few tasks with simple relationships. However, as the size of the graph increases and complexity of information transfer between tasks grows, digraphs become cumbersome to construct and decipher. Nonetheless, digraphs are central to several modeling methodologies, including CPM/PERT, which is described next.
3.1.2. CPM/PERT

Project evaluation and review technique (PERT) along with the critical path method (CPM) are one of the first time-based modeling techniques (Smith & Morrow, 1999). These techniques for project management were developed in late 1950s and have been used extensively in their improved and extended forms since then. Both of these techniques are generally used in combination and a set of steps common to both methods are presented below.

1. Define the project and all of its significant activities or tasks. The project (made up of several tasks) should have only a single start activity and a single finish activity.
2. Develop the relationships among the activities. Decide which activities must precede and which must follow others.
3. Draw the "network" connecting all the activities. Each activity should have unique event numbers.
4. Assign time and/or cost estimates to each activity.
5. Compute the longest time path through the network. This is called the critical path.
6. Use the network to help plan, schedule, monitor and control the project.

The main criticism of PERT/CPM technique is the inability to take into account real complexities of project planning and product development (Taylor III & Moore, 1980). More specifically, PERT/CPM doesn’t take into account multiple branching of projects, different teams, and feedback loops (e.g., rework).

To account for some of the drawbacks of the PERT/CPM technique, various extensions were developed. One of the extensions to PERT is known as Graphical Evaluation and Review Technique (GERT) (Moore & Taylor III, 1977), which is described in the next section.

3.1.3. GERT/Q-GERT

The main advantage of GERT over PERT is its ability to consider both deterministic and probabilistic branching of project-development activities. Furthermore, GERT allows looping such that after completing one activity it may be required to attend a previously completed activity (Moore & Clayton, 1976; Whitehouse, 1973). GERT has been used for problems with limited scope (Moore & Taylor III, 1977) and generally has not gained popularity in practice. A limiting factor in GERT is the fact that as the number of project teams and projects increases, it
becomes very complex to represent the GERT network and understand activity couplings and various loops. Q-GERT (Pritsker, 1979) has been developed to address some of these shortcomings. It offers the potential to conduct planning and scheduling activities when multiple teams and projects exist. Another differentiating aspect between Q-GERT and GERT is the inclusion of special queue nodes (hence the name Q-GERT). These nodes model situations in which queues build up before activities are serviced.

Although CPM, PERT, GERT, and Q-GERT are often used as graphical network tools for aiding product development and project management activities, simulation models have also been developed to analyze these networks. In fact, most GERT and Q-GERT network are analyzed through simulations, albeit most of them were developed in 1970s and 1980s.

3.1.4. SADT/IDEF

The Structured Analysis and Design Technique (SADT) was developed in 1970s and is a diagrammatic notation for specifying tasks, information inputs and outputs, as well as means used for tasks. According to Ross (1977), SADT provides methods for:

1. Thinking in a structured way about large and complex problems.
2. Working as a team with effective division and coordination of effort and roles.
3. Communicating interview, analysis, and design results in clear, precise notation.
4. Documenting current results and decisions in a way which provides a complete audit of history.
5. Controlling accuracy, completeness, and quality through frequent review and approval procedure.
6. Planning, managing, and assessing progress of the team effort.

Based on SADT various modeling languages were developed. Integration Definition (IDEF) denotes a family of models among which the most popular one, IDEF0 is based on SADT and represents a functional modeling language (Marca & McGowan, 2006). A process activity model discussed by Qureshi et al. (1997) uses IDEF0 for process representations used in archiving design histories. Furthermore, SADT/IDEF models were used to document complex processes both graphically and through computer tools (Park & Cutkosky, 1999). However, it has been noted that SADT/IDEF representations are hard to maintain and difficult to follow, especially
since complex processes can be several pages long when documented (Marca & McGowan, 1987).

3.1.5. Petri net

A Petri net, also known as place-transition net, is a graphical tool invented by Carl Adam Petri (Petri, 1980) and has been applied in many domains. For example, Chan and Zhang (2001) proposed a Petri net based approach for manufacturing activities in agile manufacturing environment. Tacconi and Lewis (1997) used Petri nets in a hybrid approach that allows fast design and reconfiguration of controllers for manufacturing systems. Yan et al. (2003) used Petri net approach to analyze resource assignment and design scheduling activities when a predetermined product development plan was available. Others used Petri nets in a variety of different domains (Molloy, 1982; Symons, 1987).

The structure of Petri nets consist of directed, weighted, and bipartite graph, which includes place and transition nodes. Over time, different variations of Petri nets were developed, such as colored (Zu & Huang, 2004) and boolean nets (Kansal, Acharya, & Singh, 2012). Generally, Petri nets can help describe and design discrete systems that are concurrent, asynchronous, distributed, parallel, random, and/or non-deterministic (Kansal et al., 2012). Petri nets have both graphical representations, which are useful for planning purposes, and mathematical description to model the behavior of discrete systems. A major weakness of this approach is the complexity problem, as identified by Murata (1989). More specifically, using this modeling technique, even modest-size systems can become too complex for analysis.

3.1.6. Signal flow graph

Signal flow graphs are popular in examining electrical circuits, as well as analyzing feedback control systems. Generally, signal flow graphs consist of nodes and directed branches, where nodes represent variables and branch weights represent multiplication factors for the variables. It is worth noting that signal flow graphs can be easily converted into an algebraic (matrix) representation, which makes easier to analyze the system being modeled.

Eppinger et al. (1997) used signal flow graphs to improve understanding of time-consuming design iterations during a PD lifecycle. In their representation, branches depict tasks that are being worked and branch weights include the probability and time to execute the task depicted
by the branch. The authors define graph transmission as the product of all branch weights between the start node and the finish node. The probability distribution function of the graph transmission (i.e., project completion time) is its impulse response, which can be used to calculate the expected lead time of the project. Some of the limitations of the current method are inability to capture resource constraints and queuing delays of tasks while they wait to be worked on. Also, as the number of tasks increases, the computational time for lead time calculations can become prohibitively long.

3.1.7. Design roadmap

The design roadmap (DR) framework was proposed by Park and Cutkosky (1999). It is a graph of information processing units, which specifies input and output features of various tasks, as well as dependencies between tasks. The DR was developed to address some of the shortcomings of frameworks presented above. There are two types of nodes used in the DR framework: task and feature nodes. Features are units of information or materials upon which tasks operate. As the authors mention, features are input and output entities of tasks. The DR allows modeling the following aspects of a design process:

1. Precedence relationships among tasks and corresponding input/output activities.
2. A hierarchy of objects at different levels of detail.
3. Constraint relationships among design elements.

The authors claim that the DR framework includes the minimal foundation necessary to manage collaborative and heterogeneous processes. Despite some of the improvements of the current framework, it is still cumbersome to use, especially when one needs to draw nodes and features of complex tasks. Furthermore, there is not an easy way to capture resource constraints through the DR model.

3.1.8. Precedence diagram method

Precedence diagram method (PDM) is yet another visual representation technique for depicting design and development activities and for helping identify relationships between activities and schedule them. As Wiest (1981) mentions, PDM is more flexible than PERT/CPM and can often be more accurate. Also, Wiest emphasizes that PDM is easier to draw than PERT, which is another advantage of this method. The main characteristics of PDM are the following.
1. Activities are shown as nodes and precedence relationships are represented as arrows connecting nodes.

2. More complex precedence relationships can be modeled in PDM than in CPM/PERT. For example, finish-to-finish or start-to-start relationships between activities can be represented in PDM.

3. Durations other than zero can be assigned to the precedence relationships to specify either late or early start of activities.

Although increased flexibility of the PDM is advantageous in many circumstances, it has been noted that this can create additional complexities and, in rare cases, result in loops in precedence network. Also, they can modify the concept of the critical path and affect project duration calculations. It should still be noted that the PDM does not allow feedback loops and does not take into account resource constraints, similar to the PERT/CPM technique.

3.1.9. Gantt chart

The last graphical technique described in this thesis is the Gantt chart. Although an exact date is not known for the creation of Gantt charts (Wilson, 2003), many authors believe that Gantt developed his charts during World War I (e.g., Field & Keller, 1998, p.182; Nicholas, 1989, p. 26). Gantt himself first described a version of Gantt charts in a paper published in 1903. Since then, various forms of these charts have been used for many purposes. Initially, Gantt charts helped manage and plan batch production and were less graphical and more tabular than the tools used later. After the Second World War, the charts became more popular in project management. In its most common implementation, the Gantt chart uses bar graphics to show start, finish, and duration of activities, which are placed on the vertical axis of a two dimensional chart. Time is represented on the horizontal axis. Gantt charts are popular because of the amount of information they present in a concise and easily understandable way. Nowadays, they are mainly used to facilitate scheduling. However, it should be noted that these charts are not solution techniques; rather they are useful in facilitating communication and providing a method to visually represent schedules.
3.2. Matrix Based

Matrix based PD and project management methods use a matrix to represent task dependencies, rework, as well as optimize scheduling. In this section, we describe three matrix representations: design structure matrix (DSM), work transformation matrix (WTM), and activity-activity incidence matrix. These matrix representations with accompanying frameworks are described next.

3.2.1. DSM

The DSM framework is composed of a square matrix representing tasks and their interdependencies (Browning & Ramasesh, 2007; Steward, 1981). The matrix is equivalent to the directed graph representation described in the previous section, albeit it is algebraically a more convenient way of indicating tasks and dependencies. The diagonal elements in the matrix represent the tasks, while the off-diagonal elements signify task dependencies. More specifically, the off-diagonal elements specify how the information flows from one task to another (if any). The DSM representation can be used to minimize the feedback loops by creating an upper-triangular matrix (i.e., tasks do not relate information to previously completed tasks) (e.g., Meier, Yassine, & Browning, 2007). In another implementation, Smith and Eppinger (Smith & Eppinger, 1997a) calculate expected durations of various orderings of tasks using Markov chains and the DSM representation. In their framework, each task has a fixed duration and the various orderings are used to find the minimal project completion time. Because the framework assumes fixed task durations that do not change based on the number of times they have been repeated (and are independent from task ordering sequence), it becomes computationally tractable to find the optimal ordering analytically. Later extensions of this framework take into consideration random task durations; however, this limits the number of tasks that can be analyzed in a reasonable time frame (Eppinger et al., 1997). Another extension to the DSM is the Work Transformation matrix method, which is described in the next section.

Activity-Activity Incidence Matrix is another representation that is identical to the DSM method and has been used by several researchers (Kusiak, 1993). Kusiak and Wang (1993) organize the incidence matrix in a way to simplify PD tasks. They differentiate between three matrices: uncoupled, decoupled, and coupled. An incidence matrix is uncoupled if it can be arranged so that mutually exclusive submatrices are generated (i.e., tasks in different submatrices are
uncoupled). If the matrix is in triangular form, then it is known as a decoupled matrix. Otherwise, the matrix is coupled. A triangularization algorithm is presented to reduce the number of feedback cycles, which (potentially) reduces overall PD time.

Kusiak and Park (1990) describe yet another DSM-related representation. More specifically, they construct a module-activity incidence matrix such that an element $a_{ij}$ is 1 if activity $j$ is involved in the design of module $i$. Otherwise, $a_{ij}$ is zero. The authors employ a clustering algorithm (Kusiak & Chow, 1987) to construct an uncoupled matrix, which clusters so that information exchange between the tasks of different modules is minimized.

The next section describes the Work Transformation Matrix (WTM), which is derived from the DSM and represents task durations and task rework probabilities.

3.2.2. WTM

In the WTM method (Smith & Eppinger, 1997b), the off-diagonal elements specify rework factors (also referred to as strength of dependence between tasks). More specifically, Smith and Eppinger (1997b) assume that the amount of rework at each subsequent iteration is a function of time spent on that task during the previous iteration and each task creates a deterministic amount of rework for other tasks. The diagonal elements of the WTM matrix represent task durations, which are fixed and specified in the beginning of a project. After each iteration the fraction of the task durations for the next iteration is calculated by multiplying a vector containing information on the fraction of work that needs to be done with the WTM matrix.

It should also be noted that the total work completed is correlated with the eigenvalues and the eigenvectors of the WTM rework matrix. Moreover, in the WTM representation it is easy to evaluate whether the PD process converges or not, i.e., a task creates rework that is longer than the duration of the task completed.

The WTM method also assumes:

1. All tasks are done in parallel.

This means that at every stage of PD there are resources available to work on all of the tasks simultaneously. Moreover, this assumption implies that the tasks do not need to wait for
more information from upstream tasks before they can be completed. Therefore, this assumption can be true in very limited (idealized) circumstances.

2. Rework is a function of the amount of work done in the previous iteration and the WTM matrix values stay the same over the duration of the project.

It is generally accepted that the duration of the tasks after each iteration decreases (Smith & Tjandra, 1998). However, the amount by which task rework durations decrease after each iteration can be different, which is not accounted by this method (i.e., the matrix values stay the same).

3.3. Analytical

Analytical models are mathematical models that have a closed form solution, i.e., the solution to the equations used to describe changes in a system can be expressed as a mathematical analytic function. Most analytical methods aim to provide project/product managers an optimal task schedule. Some of the simple analytical techniques only take into account task precedence relationships without accounting for resource constraints and/or multiple projects. Other methods decompose systems to find alternative sequences of tasks and/or predict slow or rapid convergence of task iterations. More advanced methods aid with task scheduling and handle finite resource constraints. Both mathematical programming and queuing-based techniques have been used in the past, which are described below.

3.3.1. Mathematical programming

Early attempts using mathematical programming focused on using integer programming techniques for formulation and optimization of task schedules (Krishnan, Eppinger, & Whitney, 1997b; Luh, Liu, & Moser, 1999). Later it was discovered that task scheduling was a generalization of job-shop scheduling problems, which were NP hard (Lawler, Lenstra, Rinnooy Kan, & Shmoys, 1993). Hence, many modern task scheduling techniques use heuristics to obtain near optimal solutions (e.g., Goldberg, Paterson, Srinivasan, & Sweedyk, 2001).

To account for even more general projects with multiple probabilistic routings and task iterations, stochastic project networks have been used. These projects are generally represented
via PERT networks and are NP hard (Neumann, 1990). Specialized cases of project networks, however, are not NP hard. One such example is the single machine job-shop example (Sidney, 1977). Pinedo and Schrage (1982) and Dempster (1982) provide an overview of stochastic scheduling. In 1990s, Gittins index (Gittins, Glazebrook, & Weber, 1989) was a popular concept to account for stochastic nature of problems when selecting tasks to complete. To simplify problems, one approach is to use the means of random variables to convert stochastic problems into deterministic ones (Pinedo, 2012). However, Federgruen and Mosheiov (1997) indicate that such an approach may not yield good performance. As for the deterministic case, many researchers developed methods for tackling these problems. Specifically, a Lagrangian relaxation based optimization method was developed by (Chen, Chu, & Proth, 1998; Luh & Hoitomt, 1993) that yields near optimal schedules. Belhe and Kusiak (1995) also used a mathematical optimization to find a lower bound on the completion time of hierarchically structured design activities. Belhe and Kusiak (1997) further improved their method to dynamically schedule design activities with resource constraints by decomposing the scheduling problem into multidimensional knapsack problems. While these models allow for near optimal scheduling in some situations, they fail to consider queuing delays, which can increase PD lead times significantly.

3.3.2. Queuing theory

In PD queuing delays can be created by the normal flow of tasks through a PD structure and by multiple tasks competing for limited resources. Queuing theory is well positioned to take into account these queuing effects. Queuing networks model job shop problems well and have been used in manufacturing, communications, and transportation, among other fields (Jackson, 1963). Adler et al. (1995) developed a generalized stochastic queuing network for analyzing PD time. In their model, the authors analyzed PD process of a certain chemical materials producing organization in an effort to accelerate development cycle time. The model captured project types with associated task flow patterns, resources (i.e., product and process engineers and technicians), tasks, task activity times, and task precedence constraints. The model was utilized to estimate the capacity of the organization by comparing the average work demand of projects with the availability of resources. Also, cycle time was estimated for various combinations of resources, task times, and task interarrival times. Despite promising modeling results, Adler et al.
(1995) focused on modeling only one division of one organization and the model needs to change significantly to be useful in other situations. Furthermore, as Smith and Morrow (1999) indicate, for such a model to work, there must be enough similar projects that data for probabilistic description makes sense and is available. Also, the projects being modeled need to be long enough that steady state probability description applies.

While the analytical methods described above have been proven to help improve PD processes and aid with project management, in general, they only capture a narrow set of factors that impact PD. Yan et al. (2003) mention that in order to “describe uncertainty and complexity in the PD process in detail and to analyze its behavior accurately” simulation models are more appropriate. Consequently, the next section describes simulation models applicable to this work.

3.4. Simulation Models

According to Axelrod (1997), simulation is a third way of doing science. “…A simulation generates data that can be analyzed inductively. Unlike typical induction, however, the simulated data comes from a rigorously specified set of rules rather than direct measurements of the real world” (pp. 24-25). To develop a simulation model, usually designers and researchers introduce various simplifications. Part of these simplifications (and assumptions) are imposed by the choice of simulation paradigms (Behdani, 2012). Lorenz and Jost (2006) mention that each simulation paradigm is characterized by a set of core / fundamental assumptions and underlying concepts. Meadows and Robinson (1985) further explain that “every modeling method is itself based on a model of how modeling should be done.” Hence, the choice of the modeling paradigm affects how the real world situation is represented and what results can be expected from the simulation model. Next, three modeling paradigms applicable to PD are discussed and compared.

3.4.1. System dynamics

System dynamics is a field of study that was developed in 1950s by Jay Forrester. To capture complexity of systems, Forrester suggested using feedback loops to represent the systems, which is also the basic building block for developing system dynamics models (Forrester, 1961). Sterman (2000) explains that a feedback loop exists any time information from one action travels through system and returns to its point of origin (not necessarily in the same form). Besides
feedback loops, stocks (or levels) and flows (or rates) and time delays are other important constructs used in system dynamics modeling. Stocks are accumulations of rates of flow. In other words, the net difference between inflow and outflow over time represents the process of accumulation (Forrester, 1968). Hence, the state of a system is described by level variables (Behdani, 2012), which implies that system dynamics models situations at an aggregate level over time. Also, system dynamics models are usually developed as continuous time models, although many computer simulations discretize the time to run the model.

Abdel-Hamid (1984) built one of the first system dynamics models on product development to specifically help manage and gain insight into the general process by which software development was conducted. This model took into account resource quantity (i.e., software developers), developer productivity, as well as effects of project targets and scope among other factors. Another important factor in product / software development that was considered in system dynamics models was distinguishing between initial work and rework. Some researchers (e.g., Abdel-Hamid, 1984; Richardson & Pugh, 1981) implicitly modeled the difference between rework and initial work, while others (e.g., Ford, Hou, & Seville, 1993) distinguished between two types of rework activities (required and optional rework). Ford and Sterman (1998) later developed a system dynamics model that accounted for work iteration and dynamic concurrency constraints. Their model was one of the most general PD system dynamics models and considered resources, process structure, targets, scope, and performance to model PD processes. The model was used to investigate the effects of process concurrency relationships, impacts of average process activity duration on overall project duration, as well as testing the impact of various management policies and development process structures.

In the next section, a more recently popularized modeling methodology, knows as agent-based modeling (ABM) is presented, along with its applications in PD.

3.4.2. Agent-based modeling

The main modeling unit of ABM methodology is the agent or individual. In ABM, designers and researchers model the agents and their behavior (North & Macal, 2007). The global system behavior emerges from the interaction of agents (Garcia, 2005). Several researchers (e.g., North & Macal, 2007; Wooldridge & Jennings, 1995) have identified a set of characteristics that agents
usually possess: (1) autonomy, (2) reactivity, (3) pro-activeness, (4) social ability, and (5) adaptivity. It is mainly because of these characteristics that ABM has been used in many different domains. For instance, Schelling (2006) developed one of the first models for modeling people as social agents. Later Epstein and Axtell (1996) modeled agents for growing entire artificial societies and observed many emergent features of societies, such as trade, wealth, disease, culture, and reproduction among others. ABM simulation applications also include modeling ancient civilizations (e.g., Kohler, Gumerman, & Reynolds, 2005), the spread of epidemics (e.g., Huang, Sun, Hsieh, & Lin, 2004), supply chains (e.g., Fang, Kimbrough, Pace, Valluri, & Zheng, 2002), the stock market and others, including manufacturing and insurance operations, electric power markets, civil disobedience, and command and control military operations (North & Macal, 2007).

ABM methodology has not been popular yet in simulating PD processes. However, several researchers attempted to develop project management and product development ABM simulations with varying levels of success. For example, Garcia (2005) described a simulation model in which firms competed for consumers based on an innovation strategy of manufacturing either incremental and/or innovative products. Garcia found that firms could be successful by building products to satisfy the demands of a niche customer group rather than focusing on the demands of the “average” customer. Zhang et al., (2009) also used ABM approach to evaluate and improve organizational planning in PD projects. Specifically, a model of team members as intelligent agents was developed by specifying agents’ attributes, behavior protocols, and knowledge. Furthermore, an environment model was created, which captured design tasks, product information, and resources. Using the model, the authors investigated the impact of team members’ skill configuration on project completion time and analyzed a job rotation strategy to save human resources. Furthermore, task scheduling behavior, human reliability, and various organizational structures were investigated with the simulation model.

Another ABM simulation developed by Zhang et al., (2012) focused on scheduling tasks in the PD process. The authors modeled agents’ scheduling behavior by analyzing scheduling heuristics of the designers and developing utility functions. The main factors included in the utility function were task urgency, task importance, individual preference, and recovery cost. The authors mentioned that the agent’s simulated behavior is close to the real working process of
developers. Nonetheless, constructing a valid utility function can be challenging. As Zhang et al. (2012), mention, “The algorithms to calculate utility function should be deeply explored in order to reflect the diversity in the agent behavior.” In a different model, Zhang et al. (2013) expanded on their previous model and differentiated between manager agent and designer/developer agents to schedule tasks and resolve resource conflicts. Using a case study simulation approach, the authors were able to shorten lead time of a project.

Other ABM simulations related to PD include Panchal’s (2009) model of different entities for a mass-collaborative PD environment and Martinez-Miranda and Pavon’s (2011) model to create a virtual team by using real world data of team members and specifying a set of tasks.

While all of the agent-based models presented above are useful in some aspects of PD, they are all focused on modeling individual agents very accurately and representing interactions between them, which are usually challenging to accomplish. It is often much easier to focus on the actual PD process rather than on agents and their interactions. One simulation paradigm that focuses on modeling processes rather than agents is the discrete-event simulation (DES) method, which is discussed in the next section.

3.4.3. Discrete-event simulation

DES models use events to represent real-world systems (Altiok & Melamed, 2010). Each event occurs at a particular time and changes the state of the system (Banks, Carson II, Nelson, & Nicol, 2014, p. 91). While real world systems function in continuous time, generally, events that change the state of the system only occur at discrete points in time. Hence, modeling events that cause the system to change its state is sufficient to represent the system. Moreover, since no change in the system occurs between events, the simulation scans the system from one event to another (without scanning the time between events). DES has evolved dramatically since 1960s. In fact, Babulak and Wang (2008) identify four generations of simulation products. The latest generation has been developed in late 1980s and Babulak and Wang (2008) note, these simulation packages allow ‘what-if’ analysis and flexibility to modify the model at any time. Wang et al. note (Wang, Sun, & Nooh, 1995) that overall the DES models have been used in areas such as (1) The design and evaluation of manufacturing processes, (2) Performance improvement of existing systems, (3) Establishment of optimum operational policies, and (4)
Development of an algorithm to support production planning and scheduling. The list of domains that DES has been applied to is extensive and includes human supervisory control (Mkrtchyan, 2011), hospitals (e.g., Ceglowski, Churilov, & Wasserthiel, 2006), finance (e.g., Boyle, 1977), air traffic control (e.g., Kapralov & Dyankova, 2012), and portfolio management (e.g., Morales et al., 2011) among others.

There are several ‘worldviews’ for implementing a DES model. The most commonly used approach focuses on processes that describe the lifecycle of events through the system (Banks et al., 2014). This formalization is known as process-interaction approach. Other DES modeling methods include event-scheduling (Cassandras & Lafortune, 2008) activity scanning as well as state-transition models, such as Markov Chains (Miller, Baramidze, Sheth, & Fishwick, 2004). All of these DES simulation strategies can be used to develop a PD model, with varying levels of fidelity. For example, Adler et al. (1995) utilized a DES technique to model workflow management. Specifically, the PD organization was represented as a stochastic processing network, which consists of a collection of “workstations,” which in turn are composed of servers (i.e., employees). The “workstations” process tasks, which are allocated to appropriate servers. Interarrival times of tasks as well as task processing times are subject to statistical variability. The authors used the model to conduct capacity analysis (e.g., identify under-staffed resources, determine project bottlenecks) and cycle time analysis (e.g., project completion time).

Browning and Eppinger (2002) also utilized a DES method to assess the impact of process architecture on PD schedule and cost. The authors accounted for several PD attributes, such as design/development iteration, activity overlapping, process duration and cost, and improvement curves. The model was used to generate PDFs describing the cost and duration of an unmanned aircraft vehicle (UAV) development project. From this data, probabilities of cost and schedule overruns can be calculated.

Another similar DES model was developed by Cho and Eppinger (2001). In their DSM-based model, the authors model stochastic task durations, task iterations and precedence constraints, as well as basic resource constraints. The simulation model was utilized to estimate the project completion time of the same UAV project that Browning and Eppinger (2002) used for their work. Later Cho and Eppinger (2005) expanded on their work to account for varying levels of task rework after each design iteration and proposed a heuristic for scheduling tasks in a
stochastic and iterative resource constrained environment. As in the previous model, project completion times for different scenarios were projected.

Other models were developed by Bhuiyan et al. (2004) to examine task overlapping and functional interaction impact on development time and by Levardy and Browning (2009) to estimate time and cost, as well as project performance and risk. Also, Abdelsalam and Bao (2006) present an optimization framework to determine a near optimal sequence of tasks to be implemented, so that the total project completion time is minimized. As the authors note, the model yields a ‘robust’ solution rather than an optimal one.

Martin and Raffo (2000) note that some of the advantages of queuing-based DES models include intuitiveness, close resemblance to the actual PD process that is being modeled, ability to easily represent task dependencies, parallel processing, stochastic nature of PD, and ease of incorporating attributes to describe various events. In the next section, the three simulation paradigms are compared.

3.5. Simulation Paradigm Comparison

The three paradigms use different constructs for modeling complex systems. Hence, each paradigm has both advantages and disadvantages. While in one situation one method can be optimal, in a different scenario another paradigm can be more useful and appropriate. Therefore, it is important to consider the major differences of these simulation methods. Table 17 summarizes the key characteristics of each paradigm, which were adapted from Behdani (2012).

One can infer from this table that agent-based modeling is a bottom-up modeling methodology, while system dynamics and DES approaches are considered top-down approaches. This fact, however, does not mean that the same phenomenon cannot be modeled with all three approaches. In fact, Borshchev and Filippov (2004) investigate correspondence between DES, SD and agent-based methodologies. They show that in many cases it is possible to convert SD and DES models to an agent-based model.

Different modeling paradigms also are used for modeling problems that require various abstraction levels (Figure 2). Specifically, SD methodology is best suited for modeling less details and more macro level observables, while agent based is generally utilized to model low abstraction levels and more details (however, can also be used for middle and high abstraction
levels). DES on the other hand is best positioned to model middle abstraction levels with medium details. The chart below shows how the three modeling paradigms are positioned based on the abstraction level these approaches usually capture (Borshchev & Filippov, 2004).

**Table 17: Key characteristics of modeling paradigms.**

<table>
<thead>
<tr>
<th>System Dynamics</th>
<th>Agent-based Simulation</th>
<th>Discrete Event Simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>System-oriented; focus is on modeling system observables</td>
<td>Individual oriented; focus is on modeling the entities and interactions between them</td>
<td>Process-oriented; focus is on modeling the system in detail</td>
</tr>
<tr>
<td>Homogenized entities; working w/ average values</td>
<td>Heterogeneous entities</td>
<td>Heterogeneous entities</td>
</tr>
<tr>
<td>No representation of micro-level entities</td>
<td>Micro-level entities are active entities that can interact w/ others and make decisions</td>
<td>Micro-level entities are passive objects that move through a pre-specified process</td>
</tr>
<tr>
<td>Mathematical formalization is in “stock &amp; flow”</td>
<td>Mathematical formalization is by “agent and environment”</td>
<td>Mathematical formalization is w/ “event, activity, and process”</td>
</tr>
<tr>
<td>Handling of time is either continuous or discrete</td>
<td>Handling of time is discrete</td>
<td>Handling of time is discrete</td>
</tr>
<tr>
<td>System structure is fixed</td>
<td>System structure is not fixed</td>
<td>Process is fixed</td>
</tr>
</tbody>
</table>

**Figure 2: Modeling paradigms with corresponding abstraction levels.**

While researchers (Garcia, 2005; Sweetser, 1999) generally mention that agent-based paradigm is a more flexible approach to model complex systems, it should be noted that the flexibility does...
not come effortlessly. In an agent-based model, the behavior of the complex system emerges from the interaction of agents. Hence, the nature and interaction pattern of the agent needs to be well known and precisely modeled. However, obtaining in depth information about agents and their interactions is not easy. Specifically, in a PD project, developers are represented as agents, which means that developers and their interaction patterns need to be well known and characterized. This requires the modeler to collect a lot of information, which almost certainly will vary from one project/organization to another, hence, making the model less applicable and cumbersome to use. Instead, it is preferable to use a DES approach and represent developers as workstations and prescribe each developer unique parameters to capture most important PD related factors, as well as increase applicability of the model. Besides, in PD projects process architecture plays a significant role on how the overall PD project is conducted (T. Browning & Eppinger, 2002) and given DES paradigm’s ability to easily capture complex processes, a DES approach was utilized in this work.

To show the different parameters that the past simulation models representing the three different paradigms capture with respect to PD, the table below was created. Table 18 depicts which PD attributes (identified in the previous chapter) have been modeled using system dynamics, discrete event, and agent based simulation paradigms. The key attributes corresponding to each modeling paradigm are also presented.

In system dynamics models, such as the ones developed by Ford and Sterman (1998) and Abdel-Hamid (1984), most of the PD attributes are partially taken into account. The fact that the attributes are not fully captured mainly stems from the high abstraction level that SD models have. For example, Ford and Sterman (1998) modeled tasks as having the same duration and task rework could occur but not partially. Similar limitations exist with other attributes as well, e.g., while different task phases could be conducted in parallel or sequentially, tasks within the phase are only scheduled sequentially.

The agent-based PD models are well positioned to account for various characteristics of developers. Hence, these models can fully capture developer flexibility and learning attributes (e.g., Zhang et al., 2013). Also, task structure can be modeled well with an agent-based approach. For instance, Christian (1995) represents different task types and durations, which are pre-specified and fixed, while Zhang (2013) notes that stochastic time duration can also be used.
Furthermore, rework can also be accounted for using ABM approach, albeit with certain limitations. Specifically, while probability of a rework is accounted for (e.g., Zhang et al., 2009), it is assumed that the whole task needs to be redone. Task relatedness, such as only parallel or sequential processing is captured, however, possibility of a partial task overlap is not accounted for.

Table 18: PD attributes captured by each modeling paradigm.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Task structure</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Team structure</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developer flexibility</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Rework</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Task relatedness</td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Learning</td>
<td></td>
<td>✔</td>
<td></td>
</tr>
<tr>
<td>Team coordination</td>
<td>✔</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Attribute is partially captured  ✔ - Attribute is fully captured

In many DES models, task structure, rework, and task relatedness are fully captured (e.g., Cho & Eppinger, 2001, 2005). Nonetheless, other factors such as team structure and coordination, as well as developer flexibility are not accounted for. Learning attribute is partially modeled by specifying the amount of learning that happens when a task is repeated, however, this learning factor is not differentiated among developers and developer performance factors are not specified.
3.6. Summary

In this chapter a review of the past literature regarding product development models and frameworks was conducted to evaluate their applicability to the representative setting. Over the last decades, various frameworks and models were researched to aid the PD process. Most of the frameworks can be classified as either graph based or matrix based. While graph based methods are usually used for scheduling activities using graphical task representation, matrix based methods generally depict task dependencies using a two-dimensional matrix format. Since most PD models are concerned with optimizing task schedules to reduce development lead time, analytical optimization techniques have also been proposed to obtain (near) optimal task scheduling solutions.

For more complex PD projects, simulation models have been proposed. More specifically, agent-based, system dynamics, and discrete event simulation methods have been used to model PD processes. Each of the simulation paradigms has its advantages and disadvantages that stem from the way the PD process is represented in each model. Nonetheless, this chapter has identified that none of the existing models account for all of the PD attributes that are presented in the previous chapter. Hence, a new model has been developed, which is described in detail in the next chapter.
4. Model Development

This model employs a queuing-based DES paradigm to estimate the lead time and cost of PD processes. As mentioned in the previous chapter, the current PD process models do not capture all of the attributes necessary to successfully represent the PD process of SMEs. In the model developed in this chapter, all of these attributes (Mkrtchyan & Srinivasan, 2014) are accounted for and the required modeling variables are presented, along with the modeling constructs.

This DES model has four interconnected components as shown in Figure 3. First, the task model describes the initial tasks, including task types, duration, and dependency between tasks. Next, the task rework model represents the rework structure, which goes beyond task dependency to capture task revisions. This includes rework probabilities, i.e., probability that one task causes rework for another task, and rework impact, which indicates rework duration. The third component is the developer model, which describes attributes pertaining to developers. These attributes are the number and type of developers, team assignments, productivity level, learning curve, coordination cost, and work hours. The last component is the queue, which stores the tasks before they are serviced by the developers. The queue has an associated discipline (i.e., scheduling algorithm), which indicates the order according to which tasks are processed by the developers.

Figure 3: High-level representation of the model.
The model operates in the following way:

1. The product/project manager specifies the list of tasks that need to be completed. This includes information regarding the duration and type of the tasks. Also, the PD manager specifies task dependency, i.e., flow of information between tasks and the minimum practical duration for completing each task.

2. Rework parameters are indicated. Specifically, the PD manager uses his/her judgment to indicate how likely it is for one task to create additional work for another task (i.e., probability of rework). Furthermore, the likely duration of the newly created work (i.e., rework impact) is also specified as a fraction of the original task duration.

3. Next, the modeler inputs how many teams are involved in the project and what are the time zones and associated work hours for each of the teams.

4. The modeler also specifies the developers that work in each of the teams. To be more precise, the type of work that each of the developers can process, the developers’ performance ratings, and the learning curve factors are indicated.

5. Next, the PD manager chooses the number of simulation runs before running the model, which is important, since it gives an opportunity to choose the level of accuracy vs. simulation run time. Also, after some point there is no value in increasing the number of iterations, because the simulation results converge and increasing the runs does not change the outcome.

6. Lastly, the model simulates developers working on the specified project according to the number of specified simulation runs. For each simulation run, the project completion time and cost is calculated, along with the average, median, minimum, and maximum values.

The next section describes the modeling constructs that are used in this queuing-based DES model.

4.1. Model Constructs

To model the PD process, four main DES constructs are used. Specifically, the events representing the tasks, the arrival processes for these events, the service processes for the events, and the queuing policy are the constructs of the model described in the next sections.
4.1.1. Events

There are two general categories of events, which represent the tasks that developers work on. The first category represents the initial tasks that need to be completed. These are the tasks that the manager thinks need to be completed for a given project to succeed.

The next category of events represents the tasks that are the result of the rework caused by incomplete information at the time of initial task breakdown and planning.

4.1.1.1. Initial tasks

Initial tasks are pre-programmed in the model. Each task has its associated type and duration. Tasks can be of very different nature depending on the PD project that is being modeled. Generally, it is the project manager’s responsibility to specify the initial tasks based on the goals of the project. When compiling a list of initial tasks, task dependencies are also specified. More specifically, the degree to which tasks can be processed in parallel and whether starting one task is dependent on finishing another are specified.

4.1.1.2. Rework generated tasks

Iteration of design/development tasks is critical to any PD process (Alexander, 1964; Browning, 1999; Kline, 1985). While product quality generally improves with each successive iteration (Leong & Smith, 1996; Safoutin & Smith, 1996), it also contributes significantly to the cost and completion time of a project (Cho & Eppinger, 2001). Much of the rework is caused by changes in information and/or assumptions upon which they were initially executed. As Smith and Eppinger (1997b) mention, “Rework therefore adapts the evolving solution to account for the modified information.” Levardy and Browning (2009) specify five causes of iteration: (1) poor initial task sequencing, (2) missing initial tasks, (3) poor communication, (4) input changes, and (5) mistakes. In this model, rework is taken into account using a design structure matrix (DSM) methodology to represent the probability that a rework is required, as well as the impact that the rework will have. These concepts are explained in more detail in the Arrival Processes and Service Processes sections.

4.1.2. Arrival processes

Tasks in a DES simulation have associated interarrival time, which describes the arrival of successive tasks. Arrivals can occur at random times or at scheduled times. When at random, the
Interarrival times are usually characterized by a probability distribution (Banks et al., 2014). The arrivals can also be either independent or dependent. In this model, arrivals can be dependent based on when other tasks are serviced, which can be of the same or have a different task type. To model the dependency of tasks, information flow between tasks is specified through a DSM. Servicing a task can trigger other tasks based on the dependency DSM, which is discussed in the next sections. The newly triggered tasks subsequently enter the queue and are later serviced according to the queue discipline. Servicing a task can also cause other tasks to be unblocked, i.e., these tasks are allowed to leave the queue and be serviced. To model the blocking/unblocking, the flow of certain tasks in the queue is temporarily stopped until a condition is met (e.g., another task is serviced), as discussed in Balsamo et al. (2001). The concept of dependency is extended in this model to account for tasks that may be unblocked after a certain percentage of a different task is finished, without waiting for that task to be fully completed (e.g., dependency of initial tasks, as described below).

4.1.2.1. Initial task interarrival times

Interarrival time of initial tasks is not explicitly modeled because availability of developers and dependency between tasks dictates when the initial tasks are serviced. To model dependency between tasks, information flow from task \( i \) to task \( j \) is specified through a dependency matrix \( T \). \( T(i,j) \) is zero when there is no information flow from task \( i \) to task \( j \) and the two tasks can be processed in parallel. \( T(i,j) \) is one when there is information flow from task \( i \) to task \( j \) and the two tasks are either processed sequentially or there can be some overlap. To specify the level of overlap between tasks \( j \) and \( i \), an overlap matrix \( O(i,j) \) is used. \( O(i,j) = b \) implies that task \( i \) can be serviced before task \( j \) has been completed and. The level of overlap is \( T_{\text{duration}}(i) \cdot O(i,j) \), where \( T_{\text{duration}}(i) \) is the initial duration of task \( i \). Hence, for example, if \( O(i,j) = 1/3 \), then task \( j \) can start after 2/3 of task \( i \) is completed.

4.1.2.2. Rework task interarrival times

Arrival of tasks associated with rework are modeled using a rework probability matrix \( R_{iq}^{\text{rework}} \), which indicates probability that task \( j \) causes rework for task \( i \) during \( q \)th iteration. More specifically, the value of \( R_{iq}^{\text{rework}} \) implies whether rework is caused for task \( i \) after \( q \)th iteration of task \( j \), i.e., the value of \( R_{iq}^{\text{rework}} \) is the success probability of the Bernoulli distribution.
4.1.3. Service processes

Similar to arrival processes, service processes can also be either constant or of random duration. Also, different tasks can have different service times (or probability distributions).

4.1.3.1. Initial task service times

To indicate the duration of initial tasks, the model allows product/project managers to input their pessimistic ($p$), likely ($l$), and optimistic ($o$) estimates. Specifically, a time duration matrix $T_d(j) = (p, l, o)$ uses latin hypercube sampling (LHS) method to generate expected values for the duration of task $j$. Latin hypercube sampling has been shown to have a better convergence rate compared to a more popular random sampling method (Chrisman, 2014; McKay, Beckman, & Conover, 1979). Since the estimate of initial service times of tasks is usually carried out by project managers based on their prior experience, this model captures the 10\textsuperscript{th} percentile of the expected duration, the mode and the 90\textsuperscript{th} percentile of the expected duration of a task as $(p, l, o)$. It has been shown that the 10\textsuperscript{th} and 90\textsuperscript{th} percentiles of the expected duration are easier for humans to estimate than the 0\textsuperscript{th} and 100\textsuperscript{th} percentiles of the probability distribution function (PDF) of expected durations (Cho & Eppinger, 2001).

For each task $j$, its pessimistic, likely and optimistic duration estimates can be used to generate a PDF $f_\xi(\xi)$. Specifically, given the 0\textsuperscript{th} ($a$) and 100\textsuperscript{th} ($b$) percentiles for task durations, as well as the mode ($c$), the PDF of a triangular distribution can be constructed. In fact, given

\[
\begin{align*}
    &a: -\infty < a < \infty \\
    &b: a < b \\
    &c: a \leq c \leq b
\end{align*}
\]

one can write the PDF of a triangular distribution as follows.

\[
f_\xi(\xi) = \begin{cases} 
    0, & \text{for } \xi < a \text{ or } \xi > b \\
    \frac{2(\xi - a)}{(b - a)(c - a)}, & \text{for } a \leq \xi \leq c \\
    \frac{2(b - \xi)}{(b - a)(b - c)}, & \text{for } c < \xi \leq b
\end{cases}
\]

To find $a, c, b$ from known estimates of $p, l, o$, the following system of equations has been derived.
\[
\begin{align*}
(1 + z)^2 + (1 + w)^2 &= 10 \\
\frac{1}{l-o}z(z+1) &= \frac{1}{o-l}w(1+w)
\end{align*}
\]

Using a publicly available library for numerical analysis and taking into account initial conditions (i.e., \(z > 0, w > 0\)), the values of \(z\) and \(w\) can be computed for any estimates of \(p, l, o\). Next, \(a = o - \frac{1}{z}(l - o)\) and \(b = p + \frac{1}{z}(p - l)\), while \(c = l\). The proof is presented in Appendix C.

4.1.3.2. Latin hypercube sampling

To generate task duration samples when running the simulation model, the LHS method is used. LHS is a type of stratified sampling (Keramat & Kielbasa, 1997; McKay et al., 1979) and it operates the following way when sampling data for \(K\) tasks. The range spaces for each component of \(\xi\), i.e., \(\xi_1, \xi_2, \ldots, \xi_K\) is split into \(M\) disjoint intervals (also known as stratas) on the basis of equal probability size, where \(M\) is the number of simulation iterations. In this case, each disjoint interval has a probability size of \(1/M\). One value from each interval is selected randomly. Hence, \(M\) values are obtained for each component of \(\xi\). Lastly, \(M\) values from \(\xi_1\) are randomly combined with \(M\) values of \(\xi_2, \xi_3, \ldots, \xi_K\) to form \(M K\)-tuples. The set of \(K\)-tuples is also known as Latin hypercube sampling. Therefore, for given \(M\) and \(K\), there exist \((M!)^{K-1}\) interval combinations for a Latin hypercube sampling. Furthermore, compared to random Monte Carlo sampling method, LHS has significantly smaller sampling error of \(O\left(\frac{1}{N}\right)\) (Aistleitner, Hofer, & Tichy, 2012; Loh, 1995), while the sampling error of Monte Carlo is \(O\left(\frac{1}{\sqrt{N}}\right)\). This means that the sampling error decreases quadratically faster in the case of LHS.

To accomplish the above-described process, an easy to implement method has been proposed, which uses inverse Cumulative Distribution Function (CDF) (Keramat & Kielbasa, 1999; Wyss & Jorgensen, 1998). First, the interval \([0,1]\) is divided into \(M\) intervals. Next, the midpoint of each interval is determined. Afterward, using inverse CDF, values corresponding to task
durations are extracted. Specifically, for a triangular distribution, random variates can be generated as follows knowing the value of \(v \in [0,1]\).

\[
\begin{cases}
    \xi = a + \sqrt{v(b-a)(l-a)}, & \text{when } 0 < v < \frac{l-a}{b-a} \\
    \xi = b - \sqrt{(1-v)(b-a)(b-l)}, & \text{when } \frac{l-a}{b-a} \leq v < 1
\end{cases}
\]

\(M\) samples are generated this way for each component of \(\xi\), which are used to specify task durations for \(K\) tasks during \(M\) simulation runs. Because this method chooses the midpoint for each of the \(M\) intervals, the method is referred to as median Latin hypercube sampling.

This process is shown graphically in Figure 4 using a CDF of a triangular distribution for generating 5 samples for the first task (i.e., the first component of \(\xi\)). The interval \([0,1]\) is divided into 5 regions. The midpoint of each interval is selected and the corresponding \(\xi\) values are calculated.

![Figure 4: Graphical representation of median Latin hypercube sampling method.](image)

### 4.1.3.3. Rework service times

Rework service times are modeled as fractions of initial service times. For example, if task \(i\) requires a rework, its duration is modeled as a fraction of the initial task, i.e, \(S_{i}^{\text{rework}} = k_{i}^{\text{rework}} S_{i}^{\text{init}}\).
$S_i^{\text{initial}}$, where $S_i^{\text{initial}}$ is the initial duration of task $i$ and $k_{iq}^{\text{rework}}$ is the fraction of work that needs to be reworked for task $i$ during $q$th iteration. The values of $k_{iq}^{\text{rework}}$ are extracted from a three-dimensional matrix $K_{iq}^{\text{j.rework}}$, which also includes information about which task $j$ causes rework for task $i$ during $q$th iteration.

4.1.3.4. Factors impacting service time

While service times are generally extracted from a PDF, they can be further impacted by a variety of factors. Some of the major factors impacting service times are the learning curve, individual developer performance variations, coordination cost, and developer work hours.

1. Number of servers

This model uses parallel servers that can be arranged in different groups. The number of groups is denoted by $N_F$ and the number of servers per each group $F$ by $N_D^F$. Hence, the total number of servers is $\sum_F N_D^F$. Servers basically act as processing units. When a task arrives, the server processes the task according to its service time and server characteristics. It should be noted that servers have varying characteristics (e.g., performance ratings, types of tasks they can service, availability hours) and they do not always have to work in parallel. For example, if two servers process tasks that are dependent, then one server has to wait until the other server finishes the task. This occurs despite the fact that both servers, in theory, can work in parallel. Nonetheless, the number of servers can greatly impact the total project completion time, since greater number of servers can process tasks faster. Another important characteristic describing the servers is their type.

2. Server type

Servers can have different types ($D_t$), which implies that tasks that servers can process are of different types. It is possible that a group of servers will have the same type but in this model it is more common for each group to have servers of multiple types. Furthermore, a server can have more than one type, which allows that server to process more than one type of tasks. For example, a server can process software tasks, as well as hardware tasks. Since all the tasks have their appropriate types, it is required that all task types have at least one matching developer with
the same type. Otherwise, a task will never be processed, if a developer with the same type does not exist.

3. Performance level

It is often the case that servers can process the same task differently, which can impact task processing time. To take this into account, the model allows the project manager to account for performance differences by specifying the performance level \((P_D)\) for each server. Specifically, the model assumes a default average performance level of 1. If a server is assumed to have better than average performance, the manager can increase the performance level by increasing the number, e.g., the server with a performance level of 1.5 implies that tasks will be completed 50% faster compared to a server with a performance level of 1.

4. Improvement curve

Most tasks get faster with practice and as Ritter and Schooler (2001) the rate and shape of improvement is fairly common across tasks. The improvement curve is usually associated with the “power law of practice” (Newell & Rosenbloom, 1981). It states that the logarithm of the time needed to complete a task decreases linearly with the logarithm of the number of times the task has been repeated. In this model, the improvement effect is accounted by introducing two variables: \(L_D\) and \(L_{T}^{\text{min}}\). \(L_D\) represents the percentage improvement when the same task is repeated by server \(D\). For example, if a task takes 200 time units to complete, next time the same server with \(L_D = 10\%\) works on the same task, the model will assume that the service time of the task has been reduced by \(10\% \times 200 = 20\) time units. The next time the task is attempted, the service time will be reduced further by \(10\% \times (200 - 20) = 18\) time units. However, as the power law states there is a minimum duration of how long it can take to complete a task, otherwise after repeating the task for many times its service time will approach zero. To take this into account, \(L_{T}^{\text{min}}\) specifies a lower limit for the duration of task \(T\).

5. Server availability

While most DES models assume that servers are always available to process a task after its arrival, in this model servers can also be unavailable to service tasks. This means that one or multiple servers can stop processing tasks for a certain period of time \((D_h)\). This is accomplished
by implementing a server blocking mechanism (Balsamo et al., 2001), which allows tasks to wait in the queue until a server of the required type becomes available. In traditional implementations of blocking before service mechanisms, if a server becomes unavailable before servicing a task is complete, the job that task has received is lost. In this work, the model keeps track of the amount of service that a job has received and resumes the servicing when an appropriate server becomes available. In effect, the server has memory, unlike most DES systems that have memoryless servers. Furthermore, depending on whether the same or a different server continues processing the task, an additional time may be required to coordinate the transfer of the task from one server to another, which is described below.

6. Task coordination between servers

Lastly, the model accounts for additional service time (as a function of $\Delta \tau_i^{cross}$) needed to process tasks when they are transferred between servers. This model assumes that when a server that is processing a task becomes unavailable and another server of the same type can continue the task, an additional service time is added to the remaining service time of the task $i$. It should also be noted that if the same server resumes processing a task after it was unavailable for some time, additional coordination time is not required.

4.1.4. Queue

Queue serves as a temporary holding place for tasks before servers process them. However, besides serving just as a storage for the tasks, the queue has a discipline, which describes the logical ordering of tasks in a queue. It determines which task will be serviced first when a server becomes available (Banks et al., 2014). There are numerous queue disciplines (e.g., first-in-first-out, service in random order, shortest processing time first) that can be implemented in a DES model. In this model, service according to priority queue discipline is implemented, which allocates highest priority task to the first available server of the same task type.
4.2. Mapping PD Attributes to Model Constructs

The next sections describe how PD attributes identified in Chapter 2 map to the modeling constructs presented above. Specifically, Table 19 summarizes how each PD attribute is represented in the model using the variables that describe model constructs.

<table>
<thead>
<tr>
<th>PD Attribute</th>
<th>Variables</th>
</tr>
</thead>
</table>
| Task structure     | • Task duration, $T_d^i$  
                     | • Task type & task assignments, $T_t^i$                                  |
| Team structure     | • Number of teams, $N_F$  
                     | • Number of developers per team, $N_F^D$  
                     | • Team assignments, $F_D$ |
| Developer flexibility | • Developer work hours, $D_h$  
                     | • Developer work type, $D_t$ |
| Rework             | • Rework impact, $K_{i,q}^{\text{rework}}$  
                     | • Rework probability, $R_{i,q}^{\text{rework}}$                         |
| Task relatedness   | • Task dependency, $T(i,j)$  
                     | • Task overlap, $O(i,j)$                                                |
| Learning           | • Productivity, $P_d$  
                     | • Learning curve, $L_D$ & $L_{T_{\text{min}}}$                         |
| Team coordination  | • Coordination cost, $\Delta S_{i}^{\text{cross}}$                      |

4.2.1. Task structure

The structure of the task plays a big role on how the task is processed by a developer. The main properties of the task structure are task duration, type, and task assignment to specific developer(s), if any. The model accounts for these properties by utilizing task duration values, $T_d^i$ for each task $i$. As explained earlier, the pessimistic, likely, and optimistic duration estimates for task durations are used construct a triangular PDF for each task. Afterward, using latin hypercube sampling method, samples are generated from the PDF and used in the model simulation.
Task type is captured through $T_i^j$ variable, which assigns task types to each task $i$. The model uses the assigned task types to match the tasks with appropriate developers, i.e., developers who can service tasks of the appropriate type. The types of tasks are user defined and can be as broad as specifying a field (e.g., electrical engineering, mechanical engineering) or very specialized (e.g., quadrotor stability control system engineering).

Lastly, if required to assign a task to specific developer(s), the task’s type can be specified in such a way that it only matches the specific developer(s) that the task needs to be assigned to. Furthermore, this way of specifying task assignments does not require the introduction of additional modeling variables.

4.2.2. Team structure

The number of teams and developers per team, as well as developer assignments to a specific team are used to capture the properties of the team structure attribute. Since developers are represented as servers in this model, the number of server groups ($N_F$) is considered to be the number of developer groups working on a project. Furthermore, the number of servers ($N^D_F$) in each group $F$ represents the number of developers that are assigned to that group. Lastly, for each developer $D$, $F_D$ specifies the team to which the developer is part of.

4.2.3. Developer flexibility

The two important characteristics of developers’ flexibility are the ability to have flexible work hours and work on tasks of varying types. To capture developers’ work hours, the model uses the variable $D_h$ to specify availability of servers, i.e., developers. In this model, $D_h$ indicates work hours of each of the developer teams. The choice of not modeling individual developer’s work hours stems from the fact that in SMEs interviewed for this work, most of the time developers in each team had very similar work hours. Hence, without losing rigor and taking into account practicality aspects of specifying work hours for each developer individually, this model accounts for work hours of developer teams. This is especially useful when modeling a distributed product development process that includes teams in multiple times zones/countries. Knowing work hours of different teams helps take into account task distributions between developers working in different time zones.
Next, developers' ability to work on different task types can be captured easily by recalling that each task \( i \) has an associated task type \( T_i \). Hence, for each developer this model indicates which task types \( (D_t) \) is developer \( D \) eligible to service. Since all the tasks have their appropriate types, it is required that all task types have at least one matching developer with the same type. Otherwise, a task will never be serviced, if a developer with the same type does not exist.

4.2.4. Rework

As rework is one of the fundamental aspects of PD, it is important to capture the essential aspects of how the rework occurs and what impact it has on the PD process. As described in the Arrival Processes section, the rework probability matrix \( R_{ij}^{\text{rework}} \) indicates probability that task \( i \) causes rework for task \( j \) during \( q\text{th} \) iteration. The impact the rework has is described in the Service Processes section. Specifically, \( K_{ij}^{\text{rework}} \) indicates the fraction of work that needs to be reworked for task \( j \) during \( q\text{th} \) iteration, which has been caused by task \( i \). These two three-dimensional matrices capture the essential aspects of rework. While a project manager needs to populate the matrices, which in some circumstances can be time consuming, it is necessary to strike a balance between the amount of information captured, the model's practicality and ability to capture PD processes reasonably well.

4.2.5. Task relatedness

The flow of information between tasks is accounted for through the use of the dependency matrix \( T(i,j) \). As explained earlier, the value of \( T(i,j) \) can be either zero or one, indicating whether there is information flow from task \( i \) to task \( j \). If there is information flow between tasks, then matrix \( O(i,j) \) specifies the level of overlap between the two tasks \( i \) and \( j \). The values of \( O(i,j) \) specifies the fraction of the duration of task \( i \) after which task \( j \) can be serviced.

4.2.6. Learning

This model captures the learning attribute through performance and learning curve variables. More specifically, varying levels of developer experience, knowledge, and commitment can impact their performance, which further impacts task service times. The performance level, \( P_D \) variable introduced earlier accounts for performance differences between developers. For each developer \( D \), the modeler specified his/her performance level.
The learning curve is approximated with the improvement curve described in the Service Processes section. The improvement that occurs after repeating the same task is assumed to be $L_D$ and can be different for each developer $D$. Moreover, $L_T^{\text{min}}$ specifies an upper limit on the maximum improvement that can occur. More specifically, $L_T^{\text{min}}$ is a fraction of the original task duration, which is the minimum amount of time needed to complete task $T$. If the projected task duration with the learning curve and performance benefit is lower than the fraction of the original task duration indicated by $L_T^{\text{min}}$, then the model uses the fraction $L_T^{\text{min}}$ of the task duration in its calculations.

### 4.2.7. Coordination cost

Lastly, the model also takes into account coordination cost between developers. The coordination cost refers to the amount of extra time that one developer needs to get accustomed to a task that another developer had been working on. The coordination cost is assumed to be zero for the same person, i.e., when a developer continues his own work there is no extra coordination cost that should be taken into account. Furthermore, the coordination cost is higher for developers that work in different teams and/or in different sites (Cramton, 2001b; Grinter, Herbsleb, & Perry, 1999; Kraut, Fussell, Brennan, & Siegel, 2002) than for developers working side by side in the same team/site (e.g., co-located developers). The model accounts for the coordination cost by increasing the service time of task $i$ when a developer from a different team previously worked on the same task. This additional service time depends on $\Delta S^\text{cross}_i$, which is known as the cross-site coordination cost for task $i$ and it is specified as a percentage of original task duration. The additional service time is calculated in the following way:

$$\text{Additional service time} = \Delta S^\text{cross}_i \cdot T_{d,\text{completed}}^i,$$

where $T_{d,\text{completed}}^i$ is the completed duration of task $i$ that another developer has worked on. For example, if $\Delta S^\text{cross}_i = 0.2$ (i.e., 20% cross-site coordination cost) and the total duration of task $i$ is 40 units, while $T_{d,\text{completed}}^i = 10$ (i.e., $1/4$th of task $i$ has been completed by a different developer), then $\Delta S^\text{cross}_i \cdot T_{d,\text{completed}}^i = 0.2 \times 10 = 2$ units of additional service time will be added to the remaining 30 units of task time when task $i$ is passed to a developer in a different team.
4.3. Task Scheduling

To allocate tasks to appropriate developer, tasks in the queue, as well as developers are prioritized. The heuristics that are used to assign priority levels to the tasks are based on task criticality and task duration. Specifically, the pseudo code below represents task prioritization algorithm.

\[
\text{For each task } i \{ \\
\text{Assign initial priority level of } Pr_i = \sum_j T(i, j) \\
\text{For each new rework task } k \{ \\
Pr_i = Pr_i + T(i, j_k), \text{ where } j_k \text{ is row } j \text{ corresponding to rework task } k \\
\} \\
\text{Sort tasks according to priority } Pr_i \text{ (high to low)} \\
\text{If } Pr_i = Pr_j, \text{ sort tasks according to task duration (shortest to longest)}
\]

If two developers have the same task type, it is possible that both developers can be available to service tasks. Hence, it is essential to know which developer the task should be assigned. For this reason, developers are also prioritized. The model allows to sort developers according to either performance or cost. If sorting according to performance, the highest priority task gets assigned to the developer with highest performance rating until there are no more tasks to assign and/or all developers are unavailable. The model can also sort developers according to hourly cost. This option can be useful to PD managers when considering minimizing project cost.

Hence, the tasks are scheduled in a way that the most critical tasks are assigned to the developers with highest performance rating. This simulates real world task scheduling in SMEs where tasks that have many other tasks dependent on them are critical and need to be processed sooner. Also, sorting the tasks in the queue according to task durations ensures that in case of tasks of the same priority level, the shortest task is processed first. Furthermore, prioritizing developers according to their performance level ensures that the most skilled and experienced developer processes the most critical task(s).
4.4. Model Outputs

The model is able to capture a variety of metrics that are useful for analyzing the PD process. Specifically, being a queuing-based DES model, it is easy to capture long-run measures of performance of this queuing system. Moreover, the model also captures PD specific metrics.

4.4.1. DES-based metrics

The main steady-state DES-based metrics are (a) the total and average time tasks spend in the queue, (b) total and time-average number of tasks in the queue, (c) average utilization and utilization of each server/developer. In this model, from the steady-state measures mentioned above, utilization can be used as a measure of developer workload. It is calculated as the ratio of the time the developer is busy servicing tasks divided by the total duration of the simulation. For a single server queuing system, the long run server/developer utilization ($\rho$) is equal to the average event arrival rate ($\lambda$) divided by the average service rate ($\mu$).

$$\rho = \frac{\lambda}{\mu}$$

For the queuing system to be stable, the arrival rate must be less than the service rate, i.e., $\lambda < \mu$. If the arrival rate is greater than the service rate, then $\rho = 1$. In real-world situations, this can happen when developers have more tasks than they can service. One way to alleviate the saturation of servers is to increase the number of developers. Since real-world PD projects usually have multiple developers working in parallel, the notion of average developer utilization can also be computed, i.e., the simple arithmetic mean of the developers’ utilization.

In this model, average task wait time in the queue is also calculated. To find the average time tasks spend in the queue, we define $W_1^Q, W_2^Q, ..., W_N^Q$ to be the time each task spends in the queue, where $N$ is the number of arrivals during $[0, \tau]$. Hence, the average time spent in the queue per event will be:

$$\bar{W}_Q = \frac{1}{N} \sum_{i=1}^{N} W_i^Q$$

As $N \to \infty$, $\bar{W}_Q \to w_Q$, where $w_Q$ is the steady-state time spent in the queue. For stable queuing systems, $w_Q$ must be bounded, otherwise wait times will grow indefinitely.
Similarly, let \( r_u \) denote the total time during \([0, r]\) in which the queue contained exactly \( u \) tasks. The time weighted average number of tasks in the queue is defined by:

\[
\bar{L}_Q = \frac{1}{r} \sum_{u=1}^{\infty} ir_u = \frac{1}{r} \int_0^{r} L_Q(t)\,dt
\]

As \( r \to \infty, \bar{L}_Q \to L_Q \), where \( L_Q \) is the long-run time-average number of tasks waiting in queue.

4.4.2. PD specific metrics

Besides these DES metrics, the model also captures the total time it takes developers to complete all the tasks. Note that due to the ability to process tasks in parallel; the total time to service the tasks is generally less than the sum of times each developer spends on servicing tasks allocated to him/her. Assuming \( T_g \) indicates the time when the last task in the queue was serviced by a developer during \( gth \) run of the model, the following metrics are calculated:

- Minimum project completion time: \( \tau_{min} = \min_g (\tau^g_L) \).
- Maximum project completion time: \( \tau_{max} = \max_g (\tau^g_L) \).
- Average project completion time: \( \tau_{av} = \frac{1}{G} \sum_{g=1}^{G} \tau^g_L \), where \( G \) is the total number of simulation runs.
- Standard deviation of project completion time: \( \tau_{SD} = \sqrt{\frac{1}{G} \sum_{g=1}^{G} (\tau^g_L - \tau_{av})^2} \).

Median value of project completion time: \( \tau_{med} = \text{value of } \tau_{med} \text{ for which } P(X \leq \tau_{med}) = P(X \geq \tau_{med}) = \frac{1}{2} \), where is \( X \) is the random variable representing project completion time after each simulation run.

Furthermore, knowing hourly rate associated with each developer, the model calculates the total cost for completing all of the tasks. Specifically, for each developer \( D \), his/her utilization, \( \rho_D \) is multiplied by the total simulation time, \( \tau^g_L \) and further multiplied by the hourly rate of the developer \( (H_D) \). For each simulation run \( g \), \( \tau^g_D \) is defined as the time spent by developer \( D \) on servicing tasks. Hence the following cost-related metrics are calculated:

- Minimum project cost: \( v_{min} = \min_g (\tau^g_D \cdot H_D) \).
- Maximum project cost: \( v_{max} = \max_g (\tau^g_D \cdot H_D) \).
Average project cost: \( \nu_{av} = \frac{1}{G} \sum_{g=1}^{G} (\tau_{D}^g \cdot H_D) \), where \( G \) is the total number of simulation runs.

Standard deviation of project cost: \( \nu_{SD} = \sqrt{\frac{1}{G} \sum_{g=1}^{G} (\tau_{D}^g \cdot H_D - \nu_{av})^2} \).

Median value of project cost:

\[
\text{value of } \nu_{med} \text{ for which } P(X \leq \nu_{med}) = P(X \geq \nu_{med}) = \frac{1}{2},
\]
where is \( X \) is the random variable representing project cost after each simulation run.

Taking into account hourly rate of developers is especially useful when considering distributed product development teams in different countries. In most cases there can be substantial pay differences between developers working in different regions/countries, which can significantly impact overall project cost.

These two PD specific metrics (i.e., total time and cost) are generally the most important metrics that a project/product manager tries to optimize before and during the development process.

4.5. Summary

A queuing-based DES model of PD process has been developed. The model was described in detail, i.e., required inputs, model architecture, and its operation were explained, along with model outputs. The main constructs of the model include (a) the task block, representing various tasks that need to be completed and rework of the tasks, (b) developer block, representing developers that work on the tasks, along with their respective teams, skills, work hours, and performance ratings, and (c) queue, which serves as a holding place for tasks before they are allocated to appropriate developers.

The model developed in this chapter contributes to the PD process modeling research by explicitly accounting for resource constraints (i.e., developers, teams, skill diversity, performance and learning levels) and task iterations without sacrificing its applicability in the real world. Furthermore, the model outputs a variety of metrics that are useful in assessing the PD processes. Specifically, in addition to DES-based metrics, the model calculates the average, minimum, and maximum values of project completion time and cost, as well as standard deviation and median values.
5. Model Verification and Validation

The objective of model validation is to confirm that the computerized model within its domain of applicability accurately represents the real world system consistent with the intended application of the model, while verification is concerned with building the model correctly. Specifically, model verification is defined as “ensuring that the computer program of the computerized model and its implementation are correct” (Schlesinger et al., 1979). It is often too costly and time consuming to determine that a model is absolutely valid over the complete domain of its intended applicability. Instead, tests and evaluations are conducted until sufficient confidence is obtained that a model can be considered valid for its intended application (R. G. Sargent, 2005).

As one can expect, model verification, validation and refinement is not a linear process, rather it is an iterative process (Banks et al., 2014) and can continue indefinitely by continuously recalibrating the model to characterize new systems as shown in Figure 5. Furthermore, verification and validation (V&V) is part of the model development process (R. G. Sargent, 2005), rather than a step undertaken after the model has been developed. However, some researchers (e.g., Wood, 1986) conducted V&V process after the model was already developed. Unsurprisingly, Wood (1986) mentions that performing V&V after the development process is both extremely costly and time consuming. In this work, the concurrent way of V&V has been performed. Specifically, during the development process various parts of the model were validated before continuing to further expand the capabilities of the model and implement them.
in the software simulation. As depicted in Figure 6, the cost of obtaining very high confidence level in the model can be quite high, hence, the modeler needs to strike a balance between the confidence level / value of the model and the time, effort, and resources that are spent on validating the model. Sargent (1982; 1984) notes that tests and evaluations need to be conducted until sufficient confidence can be obtained that the model is valid for its intended application.

To validate the model, a four-step process described by Smith and Morrow (1999) has been implemented. The process consists of the following: (1) Establish ‘face validity,’ (2) Apply the model to existing data sets gathered from industry, (3) Use the model to guide decision making in an experimental environment, (4) Use the model to guide the decision making in the real world practice. In order to follow these steps, the model needs to be implemented in simulation software. For this reason, a working simulation prototype was developed in MATLAB/Simulink environment, which was later expanded and converted into a software application using Objective C programming language.

![Figure 6: Model confidence vs. value / cost tradeoff.](image)

5.1. Model Implementation

In this section, prototype simulation software is introduced. This prototype software captures half of the attributes discussed in the previous section. Once the usefulness and preliminary validation
of the prototype model was established, an expanded version of the model was implemented. Both versions of the model implementations are discussed below.

5.1.1. MATLAB® prototype

The first version of the model developed in MATLAB/Simulink® environment captured many of the PD attributes. The goal was to build confidence in the MATLAB® computing environment as a tool for building a PD simulation model. Another factor for building this prototype model was to analyze model characteristics in a way that would not require significant time commitment and resources to build the model. Using a simulation language usually reduces the amount of flexibility but generally results in having fewer errors and reduces programming time. Taking this into consideration, SimEvents® engine within the MATLAB® environment was utilized to implement the model. This engine contains many of the standard components (e.g., queues, servers) required to develop a DES model.

5.1.1.1. Interface

The model consists of various blocks that are connected together to represent the architecture of the PD process. The main blocks are shown in Figure 7, while Appendix D.1 shows a screenshot of the MATLAB® simulation program. The modeler double-clicks on the blocks to change their parameters. To change the architecture of the model, the blocks need to be reconnected.

![Figure 7: Block diagram of the prototype model implemented in MATLAB® environment.](image)

Furthermore, this software implementation also involves low level C code to account for complex behavior that is hard (or impossible) to account for using the pre-existing blocks. Specifically, to account for task dependencies (Section 4.2.5), the values of $T(i,j)$ and $O(i,j)$ are
coded manually in a C file and the relationship between task processing and matrix values are described. As mentioned earlier, this prototype version of the model implementation does not capture all of the PD attributes described in the previous section. However, for simple PD projects the model is sufficient to analyze developer utilization, task wait times, and PD project completion time.

5.1.1.2. Functionality

This implementation captures half of the PD attributes presented in Table 19. To be more precise, the following PD attributes are accounted for in the MATLAB® based simulation:

- Task structure
- Rework
- Task relatedness
- Learning

Although team structure and coordination, as well as performance and developer flexibility attributes are not captured in this iteration of the model, the MATLAB® based simulation software is still able to produce results that are useful for PD managers. Specifically, using a historical data set (discussed further in the Validation Process subsection) the MATLAB® simulation is able to successfully replicate the completion time of a multistage PD project. Also, this implementation further instilled confidence that a DES paradigm can be used successfully to model PD processes and serves as an excellent test bed for testing various PD characteristics. Furthermore, this simulation engine provides an opportunity to ‘slow down’ the time and observe how tasks are moving through the system and monitor different performance characteristics over time, such as the number of tasks in the queue, percentage of busy developers, and task delays, among other factors. This is important, since at the actual speed of simulation it is impossible to analyze whether the simulation program operates correctly or not. As mentioned by Sargent (2005), this visual display of dynamic model behavior is a simulation model validation technique known as operational graphics.

Despite some advantages of the MATLAB® based simulation software, there are several limitations that prompted the development of an extended model. The drawbacks are described briefly below.
Lack of flexibility – The simulation software lacks in flexibility regarding changing the model's structure to adapt to different projects and organizations. For example, to add a developer, one needs to add an additional server block, add a filter to specify the types of tasks that the server can process and connect the server block to the queue and to other appropriate blocks. In simple situations, the modeler has to take 15 – 20 steps to add just one single developer. Also, if one needs to change the task structure, the source code needs to be manually manipulated.

Increased complexity – Due to the simple nature of the DES blocks in MATLAB®, it is difficult to implement scenarios that are somewhat uncommon in DES models. For instance, to consider the effects of rework, the simulation model needs to keep track of task iterations, rework probability \(R_{i,q}^{j,\text{rework}}\) and rework impact \(K_{i,q}^{j,\text{rework}}\). It is complicated to account for these factors in the Simulink® environment, because it requires assigning a separate variable to each task and saving it in the MATLAB® workspace to fully capture the effects of rework. As one can predict, this causes the size of the simulation model memory to grow dramatically along with the number of tasks. This can also cause the simulation to run slowly, which is the next limiting factor.

Slow simulation pace – Because of the preconfigured DES blocks and opportunity to drag and drop them to build a simulation model, the prototype model implemented in MATLAB/Simulink® runs significantly slower compared to models implemented in high level programming languages such as FORTRAN, C, or C++.

Inconvenience of use – It requires a great effort to change the input parameters and the structure of the PD process. In fact, one of the main reasons for developing different simulation software was the inability of project managers to correctly and efficiently input information regarding PD projects, as well as modify data of the projects that were already simulated. In short, the MATLAB® model is useful for prototyping/ purposes but not for testing with the end users.

Distribution difficulties – Lastly, the model requires MATLAB® environment, along with associated libraries to run the DES simulation model. Hence, the users of the model are required to purchase and install MATLAB® multi-paradigm computing environment. This makes it difficult to distribute the model and contribute to the PD process. Also, it is challenging to
engage the users to show the value of such a modeling tool when they need to first acquire a new software platform before they can run the model.

For these reasons, a decision was made to develop a more complete model, which not only addresses the limitations presented above but also extends the capabilities of the model.

5.1.2. SimLink™ software

The extended version of the simulation software was developed as a multi-platform application. Specifically, using Objective-C object-oriented programming language, desktop and mobile simulation applications were developed and named SimLink™. In this chapter, only the interface of the mobile / tablet application is presented. The Mac OS® application is described in Appendix D.2. It should be noted that both the Mac OS® and mobile applications have similar functionalities.

The mobile application is used on devices that have touch surfaces and is designed accordingly. The application described next runs on an iPad mini® device and its overview window is shown in Figure 8.

5.1.2.1. Overview

The application has three main sections: (1) Overview, (2) Developers, (3) Tasks. The overview section displays recently opened files, as well as the project file that has already been loaded (file name shown in Workspace Details section). All SimLink™ project files are transferrable between devices and have the same PDSM extension (for product development simulation model). From the overview screen, the modeler can change the main settings of the simulation (Figure 9).

The main simulation parameters that the modeler can change are the following.
• **Iterations** – One can choose to either simulate a PD project once or specify the number of times (iterations) to run the model. When simulating more than once, the application

![Figure 8: Window to adjust simulation settings.](image)

![Figure 9: SimLink™ overview screen.](image)
requires entering the number of iterations from 2 to 10,000.

- **Timeline** – At the end of simulation run, the application uses Gantt chart representation to display the tasks from the last run of the simulation. While this feature was initially used for debugging purposes, it has been used more extensively by managers to visually understand the distribution of tasks among developers. Specifically, by zooming in and out one can evaluate how tasks have been divided among developers. Both initial tasks and reworks are shown on the Gantt chart. The application allows choosing whether to show reworks separately from the initial tasks or include them with the initial tasks. Separately showing reworks can help visually identify the amount of rework, as well as help with debugging, if needed.

- **Reworks** – The application allows to choose whether to create a rework task when the same task (a) is in the queue waiting to be serviced (queue tasks), and (b) is already being serviced by a developer (running task). The rationale behind these options has to do with the ability of developers and project managers to quickly incorporate new information into the development project. Specifically, if task $A$ is waiting in the queue to be processed and the completion of task $B$ has created rework for task $A$, it is possible to incorporate the rework information into the initial task $A$ before that task will be serviced. Hence, the rework will not be necessary to achieve the desired task outcome. Similarly, if task $A$ is being processed by a developer and rework of task $A$ is created, in some circumstances, it is possible to incorporate rework task information into the task that is being processed, without having to service the rework task.

- **Logger** – The application allows logging of simulation results in a very detailed way. To be more precise, the advanced logging option allows recording of the flow of each task through the system, including time stamps of when tasks were started, completed, as well as developer and team parameters. This advanced logging is especially useful when analyzing a specific task flow or developer performance.
5.1.2.2. Developers

The application captures all of the developer characteristics described in Chapter 4. Specifically, by using the “+” icon on the upper right side of the screen (Figure 10) the modeler can add as many developers as needed. To add teams, the Manage Teams tab can be used on the upper left corner. This tab also displays another window to add teams and specify work hours (Figure 11). Lastly, in the Developers section, one can also indicate developer type(s), performance rating, learning curve, and the cost of the developer per hour of work (Figure 12). Hence, this allows capturing differences between developers, as well as assigning developers to different teams, if required.

![Developer window](image1)

**Figure 10: Developer window.**

![Manage team window](image2)

**Figure 11: Manage team window.**

5.1.2.2. Tasks

The next application section represents the tasks in the system. The window displays all the tasks along with corresponding duration estimates \( (o, l, p) \) and task types (Figure 13). To add a task,
one can use the "+" icon on the upper right side of the screen. For each task a variety of factors are indicated through the interface. Namely, task type, minimum duration, and duration estimates are specified (Figure 14). Furthermore, the modeler also needs to indicate dependencies between tasks, as well as rework parameters. The interface to capture these parameters is presented and explained in Appendix D.2.

Lastly, the next section describes the interface used to display the results.
5.1.2.3. Results

After the simulation run is over, the results are displayed in the Overview section of the interface. First, the application displays the probability distribution function (PDF) of the completion time of the simulated project (Figure 15). This is extremely useful for managers, since it shows graphically the relative probability of the project completion times. In fact, some of the interviewed managers noted that the PDF is often more useful than the average and the standard deviation (SD) values of the project completion time. This is due to the fact that the PDF can provide more information about the distribution of the project completion time than the average and SD combined. The user of the model is also given an opportunity to vary the number of bins (from 5 to 25) that are used to draw the PDF. This helps to adjust the PDF graph based on the number of simulation runs to achieve the best graphical representation, as well as evaluate the probability of the project completion time being in a certain range. The minimum, maximum, average, median, and SD of both project completion time and project cost are also shown. One can also choose to view a sample timeline showing the tasks and corresponding developers (Figure 16). A Gantt chart representation is utilized to display the tasks and one can scroll and zoom in/out to view the tasks. When tapping on a task, its duration and assigned developer are displayed. Lastly, to view and/or save the advanced log, the modeler can use the View Log option.
5.2. Model Advantages

The model was developed to fill in the gap of providing SMEs with a practical tool to assist in PD projects. As it has become clear after the development of the model, SimLink™ also
addresses many of the PD modeling related challenges identified in the literature. This sub-
section presents the advantages of the SimLink™ simulation model.

5.2.1. Holistic

The SimLink™ model accounts for all PD attributes that were identified in Chapter 2. By so
doing, the model holistically captures PD processes in SMEs. This allows capturing the impact
of different input parameters (e.g., work hours, different teams, individual difference between
developers) on overall project performance. Moreover, by varying the input parameters,
researchers have an opportunity to gain broad and insightful information into the effects of each
variable separately and analyze the more complex impact of several interrelated variables.

5.2.2. Relevance to practice

Many past PD models are developed primarily in academic context; hence, in many cases they
emphasize such attributes that are of less importance in the industry. Smith and Morrow (1999)
in their PD literature review mention that the standards for academic merit (e.g., mathematical
elegance, worst case tractability) are not valued as highly by PD managers, who value
practicality and relevance more. Smith and Morrow (1999) conclude, “Most of [the past] models
have not been applied on real problems in a manner that enabled action by important decision
makers.” For this reason, the SimLink™ model has been developed for PD managers and with
their help, hence practicality is as important of a factor in the development process as is
academic rigor. Greater applicability is also promoted by the architecture of the model.
Specifically, having several teams of diverse developers service various tasks is applicable for
many PD settings.

5.2.3. Resource consideration

While there are PD process models that account for task dependencies, as Browning and
Ramasesh (2007) mention, “[these models] could also account for resource constraints, which
will dramatically alter their results and may also alter many of their insights and
recommendations.” Ultimately, it is the allocation of specific resources to specific activities at
particular times that needs to be considered. The SimLink™ model captures this by modeling
resources (i.e., developers) and their attributes, such as performance rating, learning curve, and
developer work hours. When not accounting for resource constraints, it is possible to schedule
multiple activities simultaneously, while in the real world it will not always be possible due to limited resources. Also, explicitly taking into account resource constraints allows managers to quickly and easily observe the gaps in available resources and fill in the gaps by acquiring the necessary resources (e.g., developers with appropriate skill and performance level).

5.2.4. Focus on project level improvements

Focusing on project level improvements rather than on activity-activity optimization is another challenge in PD modeling that was identified in surveying PD models (Browning & Ramasesh, 2007). Local task optimizations, such as sequencing several tasks to allow for shorter overall completion time are often less important than project or system level improvements. By allowing the modeler to analyze various scenarios, the SimLink™ model focuses on the overall time and cost predictions of projects, hence allowing improvement of project level observables. Also, by optimizing the project on only a few parameters (e.g., resource utilization, task duration) leaves out many other parameters that are important in PD. This implies that if the parameters that are not accounted for in the optimization algorithm change, the overall optimization result will be the same, although real world project performance will very likely be different, hence rendering the overall optimization to be sub-optimal.

5.2.5. Ease of use

Greater ease of storage and integration, as well as increased maintainability and reusability of modeling frameworks was also identified by Browning and Ramasesh (2007) as barriers in creating PD models. Furthermore, Smith and Morrow (1999) note that due to academic focus on most PD models, no commercial software exists for most of them, which is an obstacle for wider use that will need to be remedied. For these reasons, the SimLink™ simulation model is easy to store and integrate into already existing PD processes. Also, for different projects, the simulation application allows creating different project files, which makes it easy to maintain and reuse models relating to various projects.

Furthermore, the application conceals most of the complexity of the model and presents the user an easy-to-use graphical interface to input the variables and observe the results. This promotes the adoption of the SimLink™ tool and encourages PD managers to use the simulation model.
more often. To further promote model adoption, the simulation model was uploaded to an online application marketplace, where the application will be available for free download.

5.3. Model Verification

To ensure that the conceptual model is implemented correctly, the primary verification techniques of walk-throughs and traces are utilized. Also, by using structured, object-oriented programming design and modular architecture, the number of software bugs and errors is reduced. According to Fairley (1976), the two basic approaches for testing simulation software are static and dynamic methods. Structured walk-through is a static technique and has been used in the verification process of the SimLink™ model. To conduct structured walk-throughs, each part of the simulation software code is examined to verify logically possible actions the system can take in different scenarios. The main goal of the static testing is to ensure that the number of logical gaps in the implementation (if any) is minimized. The dynamic testing is more elaborate and takes longer. Specifically, the behavior of various types of specific entities in the model is traced through the system to verify that the model operates correctly (R. G. Sargent, 2005). In this work, various tasks are traced through the system to ensure that the model correctly handles each task type. Moreover, input-output relationships are examined for reasonableness under different conditions. For example, one of the first tests aimed at verifying that the model correctly generates task duration estimates. To do this, only one task and one developer were modeled in the system. By specifying task duration estimates, one could predict what the PDF of the project completion time should look like. In fact, using the system of equations consisting of two unknowns and two equations (Section 4.1.3.1), the exact limits of the PDF of the project completion time can be calculated and compared to the model output. Figure 17 shows the model output for a triangular task completion time, given the following estimates:

\[
\begin{align*}
\text{optimistic} \ (o) &= 10 \\
\text{likely} \ (l) &= 15 \\
\text{pessimistic} \ (p) &= 25
\end{align*}
\]

Theoretically, the min, max, average, median, and SD of a triangular distribution that has a 10th percentile value of 10, mode of 15, and 90th percentile value of 25 is:
The detailed calculations of theoretical values are shown in Appendix E.1. The results show that the model accurately generates task duration PDFs (Figure 17).

Another technique suggested by Banks et al. (2014) and utilized in this work is the use of the debugger to monitor the simulation as it progresses. In fact, during the development of SimLink\texttrademark, not only the debugger was used to identify and fix software bugs but also to verify that the operation of servers, the queue, and flow of tasks agrees with the conceptual model. Lastly, the advanced logging tool, along with the graphical timeline were used to check task scheduling and servicing for reasonableness. Specifically, knowing sampled task durations and developer performance and learning factors, one can calculate task processing time for each developer and compare to model results.

After the simulation software was verified to correctly represent the conceptual model, the simulation software was utilized to validate the model. The validation procedure is described next.

\[
\begin{align*}
\text{minimum} &= 4.74 \\
\text{maximum} &= 31.72 \\
\text{average} &= 17.15 \\
\text{median} &= 16.70 \\
SD &= 5.56
\end{align*}
\]
5.4. Model Validation

A multistage validation method (Naylor & Finger, 1967) is applied in this work. This implies that (1) The model’s assumptions are grounded in theory, (2) These assumptions are validated empirically, and (3) The input-output relationships are compared to the real system. To accomplish the above-mentioned multistage validation, Smith and Morrow’s (1999) validation steps are implemented next.

5.4.1. Face validity

Face validity implies that the model, including its data, assumptions, and tractability appear reasonable on its face to model users and others who are knowledgeable about the PD process (Banks et al., 2014). Because potential users were involved in the construction of the model from its conceptualization (e.g., interviews with users acquiring information about their respective PD processes) to its implementation (e.g., software testing), the model inherently has high face validity. Additionally, the model assumptions were grounded in theory, which further contributes to high face validity of the model. Lastly, in this validation stage artificial data was generated to ensure that the model generates reasonable output. It should be noted that compared to model verification, in this step, the purpose is not to ensure that the model is implemented correctly, rather the goal is to validate that the conceptual model operates as intended.

5.4.2. Application to existing data sets

The second level of validation is the application of the model to existing (also referred to as historical) data sets gathered from industry. In contrast to the face validation level, the model is applied to realistic data sets and the results are compared the real world observations. Collecting historical data sets with enough information to populate the model proved to be a difficult task, since most projects are not documented as thoroughly as needed for the SimLink™ model. Nonetheless, a PD project was documented by Browning (1999). Specifically, Browning collected information from Boeing on task durations, learning factors, task relatedness, and rework. Table 20 shows task duration estimates. It should be noted that in this case the duration estimates represent the 0th percentile, mode, and the 100th percentile of a triangular distribution, rather than the 10th and the 90th percentiles used in the SimLink™ model. Also, in Browning’s work resource constraints are not accounted for, hence the tasks can be processed
in parallel to the extent allowed by task relatedness. To incorporate this into the SimLink™ model, each task type has its designated developer that can process the task once it arrives in the queue.

Table 20: Task duration estimates for a UV project.

<table>
<thead>
<tr>
<th>Tasks</th>
<th>0</th>
<th>2</th>
<th>3</th>
<th>Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4</td>
<td>1.9</td>
<td>2</td>
<td>3</td>
<td>0.35</td>
</tr>
<tr>
<td>A511</td>
<td>4.75</td>
<td>5</td>
<td>8.75</td>
<td>0.2</td>
</tr>
<tr>
<td>A512</td>
<td>2.66</td>
<td>2.8</td>
<td>4.2</td>
<td>0.6</td>
</tr>
<tr>
<td>A531</td>
<td>9</td>
<td>10</td>
<td>12.5</td>
<td>0.33</td>
</tr>
<tr>
<td>A521</td>
<td>14.3</td>
<td>15</td>
<td>26.3</td>
<td>0.4</td>
</tr>
<tr>
<td>A522</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>A5341</td>
<td>7.2</td>
<td>8</td>
<td>10</td>
<td>0.35</td>
</tr>
<tr>
<td>A532</td>
<td>4.75</td>
<td>5</td>
<td>8.75</td>
<td>1</td>
</tr>
<tr>
<td>A533</td>
<td>18</td>
<td>20</td>
<td>22</td>
<td>0.25</td>
</tr>
<tr>
<td>A5342</td>
<td>9.5</td>
<td>10</td>
<td>17.5</td>
<td>0.5</td>
</tr>
<tr>
<td>A5343</td>
<td>14.3</td>
<td>15</td>
<td>26.3</td>
<td>0.75</td>
</tr>
<tr>
<td>A5344</td>
<td>13.5</td>
<td>15</td>
<td>18.8</td>
<td>0.3</td>
</tr>
<tr>
<td>A54</td>
<td>30</td>
<td>32.5</td>
<td>36</td>
<td>0.28</td>
</tr>
<tr>
<td>A6</td>
<td>4.5</td>
<td>5</td>
<td>6.25</td>
<td>0.7</td>
</tr>
</tbody>
</table>

Next, task relatedness is specified through a DSM, which is shown in Table 21. As described in Chapter 3, non-zero (or empty) values of the DSM indicate information flow between corresponding tasks. The super-diagonal elements indicate precedence relationships, while the sub-diagonal elements indicate feedback relationships between tasks.

It should be noted that the overlap matrix $O(i,j)$ is zero for this project, since Browning did not collect information and it was assumed that partial task overlap was not applicable.

Similarly, both rework probability and impact matrices were captured and included in the model. Appendix E.2 shows both matrices. After running the model $N = 200$ times, the SimLink™ results for project completion time are the following:

\[
\begin{align*}
\text{Average} & = 151.6 \text{ days} \\
\text{Standard deviation} & = 15.9 \text{ days}
\end{align*}
\]
Table 21: Task relatedness matrix for UV project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The results are compared to Cho and Eppinger's (2001) modeling work, which also utilizes DES simulation and uses Browning's UV data set. Specifically, the results from the previous (simpler) model were:

\[
\begin{align*}
\text{Average} &= 146.8 \\
\text{Standard deviation} &= 17.0
\end{align*}
\]

The results show that the SimLink\textsuperscript{TM} model in a simple form is able to replicate the results using a historical data set. Also, the PDF of project completion time is positively (right) skewed (Figure 18), which agrees with the results from Cho and Eppinger (2001) and Browning (1999).
5.4.3. Guiding decision making in an experimental environment

The next level of validity is the use of the model to guide decision-making in an experimental environment. The goal is to demonstrate the possibility of improved decision making using the model, while not influencing real world decisions. To accomplish this, the model was used with two small size software development firms to analyze two different PD projects. These projects, along with modeling results are presented next.

5.4.3.1. Mobile software development project

This project is concerned with revising an existing system to better support the growing demands of a small firm’s corporate clients. At the time this firm used Pivotal Tracker, an agile project management tool. Some of the information about the project, such as the tasks and developers working on the project were extracted from Pivotal Tracker. The rest of the required information was collected from a team leader responsible for this project. Table 22 shows the tasks, their types, and duration estimates.

<table>
<thead>
<tr>
<th>ID</th>
<th>Short description</th>
<th>Type</th>
<th>Durations (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Migration to new data center</td>
<td>Games back-end</td>
<td>18 22 30</td>
</tr>
<tr>
<td>2</td>
<td>Store offers in secondary cache</td>
<td>Customer API</td>
<td>5 8 14</td>
</tr>
<tr>
<td>3</td>
<td>Push ads directly to customers</td>
<td>Ads back-end</td>
<td>16 25 28</td>
</tr>
<tr>
<td>4</td>
<td>Synchronize request IDs automatically</td>
<td>Games back-end</td>
<td>12 14 16</td>
</tr>
<tr>
<td>5</td>
<td>Create plan for API integration</td>
<td>API integration</td>
<td>2 3 5</td>
</tr>
<tr>
<td>6</td>
<td>Move async engine to a two server configuration</td>
<td>Ads back-end</td>
<td>6 8 9</td>
</tr>
<tr>
<td>7</td>
<td>Increase pixel size from 1 to 4 in mobile ads</td>
<td>Games back-end</td>
<td>1.5 2 3</td>
</tr>
<tr>
<td>8</td>
<td>Update API library to newer version</td>
<td>API integration</td>
<td>15 18 24</td>
</tr>
<tr>
<td>9</td>
<td>Add an option to update logs frequently</td>
<td>Ads back-end</td>
<td>4 9 11</td>
</tr>
<tr>
<td>10</td>
<td>Test APIs with clients</td>
<td>Customer API</td>
<td>18 24 36</td>
</tr>
<tr>
<td>11</td>
<td>Update API with latest revisions</td>
<td>API integration</td>
<td>3 5 7</td>
</tr>
</tbody>
</table>

Four developers were involved in this project. To be more precise, two developers from California and two developers from Armenia worked on the project as a small geographically
distributive software development team. The developers had different performance levels, learning curves, and various skills. Table 23 summarizes the main characteristics of developers, as well as typical work hours (indicated in Armenian time).

Table 23: Developer characteristics for mobile soft. dev. project.

<table>
<thead>
<tr>
<th>Developer</th>
<th>Skills / task types</th>
<th>Performance level</th>
<th>Learning curve</th>
<th>Region</th>
<th>Typical work hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>• Games back end</td>
<td>0.75</td>
<td>0.2</td>
<td>Armenia</td>
<td>9:00-20:00</td>
</tr>
<tr>
<td></td>
<td>• API integration</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>• Customer API</td>
<td>1</td>
<td>0.1</td>
<td>California</td>
<td>20:00-06:00</td>
</tr>
<tr>
<td>3</td>
<td>• API integration</td>
<td>1.3</td>
<td>0.1</td>
<td>California</td>
<td>20:00-06:00</td>
</tr>
<tr>
<td>4</td>
<td>• Customer API</td>
<td>1.4</td>
<td>0.2</td>
<td>Armenia</td>
<td>9:00-20:00</td>
</tr>
<tr>
<td></td>
<td>• Advertisement back end</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Games back end</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The remaining modeling parameters, such as task relatedness and rework matrices are shown in Appendix E.3.

Before the project was complete, the simulation model was utilized to predict project completion time. Moreover, the timeline was analyzed to identify opportunities to increase team efficiency and compare them with real world observations. The results (Figure 19) show that model predictions for average project completion time was 182.7 with SD of 25.84, while actual project completion time was 168 hours. While actual results were within one SD of model predictions, lower actual project completion time was mainly due to increased concurrency than what was accounted in the model. Specifically, the overlap matrix, $O(i,j)$ was specified as having all zero elements (i.e., tasks that exchange information do not overlap), however, in the real world developers collaborated more than anticipated by the PD manager and some overlap between tasks was present. Still the estimated results were within one SD of the actual project completion time.

One interesting observation from analyzing the timeline of simulated projects was the fact that the workload was not distributed evenly among the teams. Particularly, the US team (Team 2 in Figure 20) often had to wait for the team in Armenia to complete the tasks before they could
start. On average, the team in Armenia did more work (on this project) than the team in the US. Although it was not necessary to identify workload distribution inefficiencies for validating the model, it reveals another way the model can be useful in improving the PD process.

5.4.3.2. Network improvement project

The main goal of this software development project was to improve the advertising network for mobile games. Task durations were approximated with a project manager are shown in Table 24 below. The characteristics of five developers involved in this project are further described in Table 25. It should be noted that in this project each task was assigned to a specific developer, hence each task has one associated task type, which corresponds to only one developer. Task types are named by the first two letters of the developers' names.
Table 24: Task types and durations for network improvement project.

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task type</th>
<th>Duration estimates (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>1</td>
<td>SE</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>SR</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>SR</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>LE</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>LE</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>AR</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>MK</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>MK</td>
<td>15</td>
</tr>
<tr>
<td>9</td>
<td>MK</td>
<td>8</td>
</tr>
<tr>
<td>10</td>
<td>LE</td>
<td>4</td>
</tr>
<tr>
<td>11</td>
<td>AR</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>LE</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>MK</td>
<td>18</td>
</tr>
<tr>
<td>14</td>
<td>AR</td>
<td>4</td>
</tr>
<tr>
<td>15</td>
<td>MK</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>SR</td>
<td>5</td>
</tr>
</tbody>
</table>

The rest of the input parameters for this project are presented in Appendix E.4. Expected average project completion time was estimated to be 203.2 hours with SD of 17.3 hours. As Figure 21 shows, the PDF of project completion time appears to be bimodal. This type of information in many cases can be more useful than the model's estimation of the average completion time. The manager can also further analyze the model to understand what causes the PDF to be bimodal. For this project, actual project completion time was 195 hours, well within SD and the actual value is closer to the major (i.e., left) mode.

Table 25: Developer characteristics for network improvement project.

<table>
<thead>
<tr>
<th>Developer</th>
<th>Performance level</th>
<th>Learning curve</th>
<th>Region</th>
<th>Typical work hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>SE</td>
<td>1</td>
<td>0.05</td>
<td>US</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>SR</td>
<td>0.6</td>
<td>0.1</td>
<td>US</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>LE</td>
<td>2</td>
<td>0.2</td>
<td>US</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>AR</td>
<td>1</td>
<td>0.1</td>
<td>US</td>
<td>22:00-06:00</td>
</tr>
<tr>
<td>MK</td>
<td>1</td>
<td>0.15</td>
<td>Armenia</td>
<td>10:00-18:00</td>
</tr>
</tbody>
</table>
The modeling work presented above for two distinct projects validates that the model can be used on real world data to make meaningful predictions and, if needed, guide the decision making process. The next section describes a project, which was altered based on model’s recommendations.

5.4.4. Guiding decision making in the real world practice

The last level of validity is the use of the model to guide decision-making in real world. This is also the most valuable application of the model, since the ultimate goal of any model is to help make better decision in real world. To achieve this level of validity, the SimLink™ simulation model was utilized to predict project completion time of a project that aims to implement a new feature. Specifically, the project was concerned with updating the graphical interface of a control system before presenting it to a potential client. The aim of the PD manager was to finish the project in four days and the model was used to numerically assess the probability of not being able to finish the project on time.

The number of developers expected to get involved in the project was three, all of them located in the US. Table 26 presents the characteristics of the developers working in this project. The project manager specified that each developer works on average 6.5 hours every day on the tasks related to this project. Furthermore, hourly cost for each developer is specified the table. It should be noted that this does not signify actual hourly rate for each developer, rather the values are used to compare PD cost for various scenarios. Nonetheless, hourly rate ratios between developers corresponds to the actual pay ratios.
Table 26: Developer characteristics for new feature implementation project.

<table>
<thead>
<tr>
<th>Developer</th>
<th>Skills / task types</th>
<th>Performance level</th>
<th>Learning curve</th>
<th>Cost per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>GUI</td>
<td>0.75</td>
<td>0.05</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>Functionality</td>
<td>0.4</td>
<td>0.05</td>
<td>50</td>
</tr>
<tr>
<td>3</td>
<td>QA</td>
<td>0.7</td>
<td>0.1</td>
<td>70</td>
</tr>
</tbody>
</table>

Task types and project durations are presented in Table 27.

Table 27: Task characteristics for new feature implementation project.

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Task type</th>
<th>Durations (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>o</td>
</tr>
<tr>
<td>1</td>
<td>GUI</td>
<td>2.25</td>
</tr>
<tr>
<td>2</td>
<td>Functionality</td>
<td>2.25</td>
</tr>
<tr>
<td>3</td>
<td>Functionality</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>Functionality</td>
<td>0.3</td>
</tr>
<tr>
<td>5</td>
<td>GUI</td>
<td>1.8</td>
</tr>
<tr>
<td>6</td>
<td>GUI</td>
<td>0.4</td>
</tr>
<tr>
<td>7</td>
<td>GUI</td>
<td>0.4</td>
</tr>
<tr>
<td>8</td>
<td>Functionality</td>
<td>0.4</td>
</tr>
<tr>
<td>9</td>
<td>GUI</td>
<td>0.6</td>
</tr>
<tr>
<td>10</td>
<td>GUI</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>Functionality</td>
<td>0.4</td>
</tr>
<tr>
<td>12</td>
<td>Functionality</td>
<td>0.4</td>
</tr>
<tr>
<td>13</td>
<td>GUI</td>
<td>0.75</td>
</tr>
<tr>
<td>14</td>
<td>GUI</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>GUI</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>GUI</td>
<td>0.75</td>
</tr>
<tr>
<td>17</td>
<td>QA</td>
<td>6.5</td>
</tr>
<tr>
<td>18</td>
<td>QA</td>
<td>7</td>
</tr>
<tr>
<td>19</td>
<td>QA</td>
<td>6</td>
</tr>
</tbody>
</table>

The simulation results indicate average expected project duration of 3.5 days with SD of 0.4 days. Additionally, minimum and maximum project durations were 2.7 and 4.5 days, respectively. Given that the maximum project completion time was above 4, the PD manager decided to add a QA to the development team to speed up the process. Model predictions with an additional QA were the following: average completion time of 3.4 days and maximum project
completion time of 4.3 days. While these numbers seem to be better compared to having just one QA, the cost of completing the project also increases. Figure 22 shows project completion times and cost for each scenario. Appendix E.5 shows the remaining modeling parameters. While analyzing this project using the SimLink™, it was observed that the most likely bottleneck is not the QA, rather the developers have more work than they can complete on time for the QA to test the completed work for bugs and errors. Hence, the model was used to analyze a different scenario in which an additional developer is added to the team. Appendix E.6 shows the modeling parameters for the additional QA and the developer.

The addition of a developer not only reduces average (3.2 days) and maximum (3.9) project completion time but also the model predicts to lower the average project cost compared to adding a QA. This analysis prompted the PD manager to organize a meeting with the PD team. The team members confirmed that the QA can generally finish reviewing the tasks in a timely manner and it’s the developers that usually have work to do (mainly to revise already completed tasks). Using this feedback and model predictions, the PD manager decided to add a developer to the team rather than a QA, as they originally intended to.

After completing the project and calculating all the time that developers spent working on this project, it was found that the total time was 3.4 days. This time was above the average model
prediction but very close to it. Hence, the model not only proved to be useful in replicating real world results but also beneficial in successfully guiding PD decision-making process.

5.5. Sensitivity Analysis

When validating a model it is important to conduct sensitivity analysis to test the robustness of the model. This test analyses model outputs when model assumptions are varied over the reasonable range of uncertainty (Sterman, 2000). It is the modeler’s desire to create a model that is robust to input errors, since input parameter estimates are likely to be imperfect and prone to error. Furthermore, it is beneficial to identify parameters to which the model is sensitive, so that the users can put in more effort to correctly estimate these parameters, as well as better understand system behavior.

To conduct sensitivity analysis, it was decided to analyze the change in project completion time after independently varying three input parameters: task duration estimates, developer performance ratings, and developer learning curve values. The reason behind selecting these input parameters is the fact that the rest of the input parameters (e.g., number of developers, developer type of work, and work hours) can be estimated with more accuracy, hence there is less need to analyze the sensitivity of the model to these parameters. Furthermore, the task duration estimates, developer performance and learning curve values can have large impact on project completion time and cost.

To conduct the analysis, errors were introduced by shifting the duration estimates and changing performance and learning curve values of developers. Specifically, optimistic, likely, and pessimistic task duration estimates were changed by \( e \) percent, where \( e \) represents input error. Also, the values for developer performance ratings and learning curves were changed. The output errors were estimated by comparing original model mean prediction values with model predictions after introducing errors.

The previous section (i.e., Section 5.4.4, Guiding decision making in the real world practice) presented a project for which sensitivity analysis was conducted. First, task duration estimates (i.e., \( T^d_i \)) were varied for each task \( i \) according to the following formula:

\[
T^d_{i_e} = (1 + e)T^d_i
\]
The analysis used $e$ values from $-20\%$ to $20\%$ with $10\%$ increments. The results indicate that the model is moderately sensitive to task duration estimates. In fact, the output mean error was always smaller than the input error. The error was the smallest when the input task durations were reduced by $20\%$. The reason behind this small output error is the dependency between tasks, which can cause tasks to wait in the queue for other tasks to be serviced, hence contributing to overall project completion time. Even if tasks can be completed faster, they may need to wait in the queue, thus increasing project completion time and reducing output error. Figure 23 summarizes the results for errors in task duration estimates.

Next, developer performance values were varied by introducing $e$ error. As previously, the values of $e$ varied from $-20\%$ to $20\%$ with $10\%$ increments. The results (Figure 24) show that there is an almost linear relationship between input and output errors. Furthermore, output is not too sensitive to the different values of input error. Also, it should be noted that larger developer performance value (or positive input error) results in smaller project completion time/cost (or negative output error), hence the negative slope of the graph in Figure 24.
Lastly, developer learning curve ($L_D$) values were varied and the model outputs were analyzed. Since the values of $L_D$ were specified as percentages, the input error $e$ was introduced in the following way:

$$L_{De} = L_D + e$$

The values of $e$ varied from $-10\%$ to $10\%$ with $5\%$ increments. The results are displayed in Figure 25. The output error in this case is more sensitive to input error. In fact, a $5\%$ increase learning curve values results in completing the project $12\%$ sooner. Further increasing the learning curve values by another $5\%$ does not affect the project completion time in any significant way. The main reason is the fact that even with only $5\%$ increase in learning curve values, after repeating tasks of the same type a few times the learning curve stops improving because the minimum allowable task duration (or maximum allowable benefit from repeating a task, $L_T^{min}$) is reached. Further increasing $L_D$ will not impact the output parameters in any significant way.

Figure 24: Sensitivity analysis for developer performance rating.
Similarly, when reducing $e = -5\%$, the output error for project completion time is 20%, while the error corresponding to project cost is 24%. This is due to the fact that two of the three developers in this project had estimated learning curve values, $L_D = 5\%$. Hence, by reducing their respective values by 5% eliminated entirely the learning effect for two of the developers. This resulted in a large output shift. Further reducing the learning curve values only affected one of the developers (i.e., QA), hence did not have any substantial impact on the output values. The greater sensitivity of the model to the learning curve value can be explained by its cumulative nature, i.e., each time a task of the same type repeats its duration is reduced by a fraction of the previously calculated duration. This implies that system modelers need to pay considerable attention when setting $L_D$ and $L_T^{\text{min}}$, especially when a project contains many tasks of the same type that are assigned to a few developers.

5.6. Internal Model Stochasticity
Another important factor when validating a simulation model is the internal stochasticity. Specifically, reasonable variability exhibited in the model’s outputs is important to build confidence and validate the model. Extremely large model variability can cause PD managers not to trust the results. Similarly, extremely low variability can also be questionable since it implies no uncertainty.
To conduct stochasticity analysis, model predicted SD was compared to previous estimates of SD values. It is preferable to compare model's predictions to data from real world project; however, SD information of project completion time and cost is not available from actual projects (because in real world a PD project is completed once and is rarely repeated).

Figure 26 shows SD comparisons for the same UV project presented in the Application to existing data sets subsection of this chapter. The project completion time SD is compared for two instantiations of the model. First, a random Monte Carlo sampling method was used for generating time duration samples, similar to Browning's (1999) work. Next, a latin hypercube sampling was used, along with modeling feed forward iterations, as presented in Cho and Eppinger (2001). As can be seen in Figure 26, in both cases the model estimates standard deviation values to be very close to the values predicted by previous models. This shows that the model has the appropriate level of variability.

Additionally, the outputs of the model from different simulation runs were compared for consistency. Specifically, the model was run for 200 iterations and for three different projects and the results were compared. For each of the three projects the maximum variability in the results was 5.2%, which is sufficiently small to show that one simulation run with more than 200 iterations is satisfactory to get accurate results.
5.7. Summary

This chapter presented model verification and validation process. First, to conduct validation process, the model was implemented in MATLAB® environment and later expanded into a standalone software application (named SimLink™) for Mac OS® and iOS® devices. The SimLink™ application captured all of the PD attributed identified in Chapter 2. The functionality and interface of the iOS® application was presented in this chapter, while for information regarding the Mac OS® application the reader is referred to the Appendix. The chapter also discussed the advantages of the model, which include relevance to practice, resource consideration, focus on project level improvements, and ease of use.

Next, the verification process of the model was presented. Specifically, both walk-throughs and traces were utilized to ensure the conceptual model was built correctly. Moreover, the debugger was used to evaluate that various modeling constructs, such as the server, and the queue function properly. Once the verification process was finished, the model was validated for various scenarios.

A multistage validation technique was used in this thesis. First, face validity of the model was established. Next, the model accurately predicted project time completion results using a historical data set for an unmanned aerial vehicle example. In the next validation stage, the model was used to guide decision making in an experimental environment. Specifically, the model accurately replicated time completion results for two software development projects. In the last stage of model validation, the model was used to guide decision making in the real world practice. In this case, the model helped the PD team make a decision to add another developer to speed up the development process.

This chapter also presented sensitivity analysis of the model, which is important for evaluating model’s robustness to input errors. The analysis showed that for a specific project the model output was not too sensitive to input errors in task duration estimates and developer performance ratings. However, the model was sensitive to input errors in the learning curve. It is important to identify this sensitivity early in the modeling process, so that the PD manager can spend more time and effort to correctly identify parameters that are sensitive to input errors.
Lastly, internal model stochasticity was evaluated to ensure that the model exhibits reasonable output variability. To conduct stochasticity analysis, model predicted SD was compared to previous estimates of SD values. Also, different simulation runs were compared for consistency. The results showed that the model has an appropriate level of stochasticity.
6. Model Synthesis

Product development (PD) managers of small and medium enterprises (SMEs) have access to a variety of practical tools (e.g., Agile Bench, Pivotal Tracker) to help them coordinate the PD process. However, these tools do not have the appropriate level of granularity nor do they capture all of the key attributes that are important in PD in SMEs. Specifically, these models do not capture human resource constraints, distributed PD teams, nor do they capture flexible developer skills or work hours.

The purpose of the SimLink™ model is to assist PD managers of SMEs in improving and analyzing PD processes. Specifically, the goal is to provide managers with an easy-to-use decision support tool to make more informed decisions that are not based solely on heuristics or gut feelings, but those that are supported by rigorous calculations and meticulously estimated probabilities.

This chapter presents various uses of the SimLink™ model. These model applications are by no means all-inclusive; rather they present the most common uses of the model that were encountered during the development and validation process.

6.1. Potential Uses

The SimLink™ model can be used in three areas. First, the simulation model can be used to set schedule and cost targets for projects. Next, the model can be useful in process improvement, while taking into account developer attributes. Lastly, the model can be utilized to answer research-oriented questions regarding PD.

6.1.1. Setting schedule and cost targets

Since the model generates a probability distribution function (PDF) of simulated projects' completion time, it is easy to use this data to set schedule targets, given a risk level of not completing the project on time. In fact, in the SimLink™ mobile application it takes very little effort to estimate the probability of exceeding a specified schedule target. More precisely, by pressing on the graph of the PDF of the project completion time, the application shows the probability of the project being completed in the highlighted interval (Figure 27). The user of the model can observe the probability for different project completion intervals by changing the number of bins that are used to generate the PDF, as well as by pressing in different parts of the
Figure 27 uses the unmanned vehicle (UV) example presented earlier (Section 5.4.2) and implies that there is an 8% probability that the project completion time will be in the interval from 161.0 days to 169.8 days. Also, there is a 0.5% probability that the project completion time will be in the range of 169.8 to 178.6. Hence, this implies that there is an 8.5% chance that the project will be completed in more than 161 days. Therefore, the model can help the PD manager easily estimate the risk of project time overrun.

Besides being able to estimate the probability of a project not finishing on time, the model can also be utilized to conduct cost and schedule tradeoff analysis. The range of possible cost and schedule values can be estimated using the model. In general, by allocating more (or better) resources it is possible to speed up the development process. Figure 28 depicts many possible outcomes for various values of cost and schedule (Ahmadi & Wang, 1994). The figure also shows an efficient frontier for the project completion time. The values on the frontier indicate the shortest possible project completion times for a given cost value. As one can imagine, the cost of the project can always be increased without positively impacting project completion time, hence, the many possible outcomes that do not lie on the frontier.
The model can be used to numerically assess this schedule and cost tradeoff for different projects and various PD configurations. The mobile development project example presented in subsection 5.4.3 is used to assess project duration and cost tradeoff. To be more specific, developer performance, learning curve, and hourly rate values are changed. The duration estimate for the original scenario is 185 hours (shown with square marker on Figure 29). Two additional scenarios are considered (shown with circular and triangular markers). The developer parameters corresponding to each scenario are selected based on the developers that are available and have the same skills (albeit with different performance and learning curve values). Table 28
lists developer parameters for each scenario.

Figure 29 shows interesting relationship between project cost and durations. Specifically, in scenario 1 (represented with triangular marker), projected duration increases (from 185 hours to 225 hours) along with the predicted cost. Hence, it does not provide any improvement compared to the original scenario. This is caused by low performance ratings of developers, who also have lower hourly rates but have to work longer to complete the same tasks.

![Table 28: Developer parameters for scenario analysis.](image)

<table>
<thead>
<tr>
<th>Developer ID</th>
<th>Original scenario</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
</tr>
<tr>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
</tr>
<tr>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
</tr>
<tr>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
</tr>
<tr>
<td>37</td>
<td>38</td>
<td>39</td>
<td>40</td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
</tr>
<tr>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
</tr>
<tr>
<td>49</td>
<td>50</td>
<td>51</td>
<td>52</td>
</tr>
<tr>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
</tr>
<tr>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
</tr>
<tr>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
</tr>
<tr>
<td>69</td>
<td>70</td>
<td>71</td>
<td>72</td>
</tr>
<tr>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
</tr>
<tr>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
</tr>
<tr>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
</tr>
<tr>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
</tr>
<tr>
<td>89</td>
<td>90</td>
<td>91</td>
<td>92</td>
</tr>
<tr>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
</tr>
<tr>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
</tr>
<tr>
<td>101</td>
<td>102</td>
<td>103</td>
<td>104</td>
</tr>
<tr>
<td>105</td>
<td>106</td>
<td>107</td>
<td>108</td>
</tr>
<tr>
<td>109</td>
<td>110</td>
<td>111</td>
<td>112</td>
</tr>
<tr>
<td>113</td>
<td>114</td>
<td>115</td>
<td>116</td>
</tr>
<tr>
<td>117</td>
<td>118</td>
<td>119</td>
<td>120</td>
</tr>
<tr>
<td>121</td>
<td>122</td>
<td>123</td>
<td>124</td>
</tr>
<tr>
<td>125</td>
<td>126</td>
<td>127</td>
<td>128</td>
</tr>
<tr>
<td>129</td>
<td>130</td>
<td>131</td>
<td>132</td>
</tr>
<tr>
<td>133</td>
<td>134</td>
<td>135</td>
<td>136</td>
</tr>
<tr>
<td>137</td>
<td>138</td>
<td>139</td>
<td>140</td>
</tr>
<tr>
<td>141</td>
<td>142</td>
<td>143</td>
<td>144</td>
</tr>
<tr>
<td>145</td>
<td>146</td>
<td>147</td>
<td>148</td>
</tr>
<tr>
<td>149</td>
<td>150</td>
<td>151</td>
<td>152</td>
</tr>
<tr>
<td>153</td>
<td>154</td>
<td>155</td>
<td>156</td>
</tr>
<tr>
<td>157</td>
<td>158</td>
<td>159</td>
<td>160</td>
</tr>
<tr>
<td>161</td>
<td>162</td>
<td>163</td>
<td>164</td>
</tr>
<tr>
<td>165</td>
<td>166</td>
<td>167</td>
<td>168</td>
</tr>
<tr>
<td>169</td>
<td>170</td>
<td>171</td>
<td>172</td>
</tr>
<tr>
<td>173</td>
<td>174</td>
<td>175</td>
<td>176</td>
</tr>
<tr>
<td>177</td>
<td>178</td>
<td>179</td>
<td>180</td>
</tr>
<tr>
<td>181</td>
<td>182</td>
<td>183</td>
<td>184</td>
</tr>
<tr>
<td>185</td>
<td>186</td>
<td>187</td>
<td>188</td>
</tr>
<tr>
<td>189</td>
<td>190</td>
<td>191</td>
<td>192</td>
</tr>
<tr>
<td>193</td>
<td>194</td>
<td>195</td>
<td>196</td>
</tr>
<tr>
<td>197</td>
<td>198</td>
<td>199</td>
<td>200</td>
</tr>
<tr>
<td>201</td>
<td>202</td>
<td>203</td>
<td>204</td>
</tr>
</tbody>
</table>

Scenario 2, on the other hand, provides about 6% time improvement compared to the original scenario. However, the cost increases more than 11%. Therefore, the PD manager needs to conduct cost benefit analysis to understand how much 6% time improvement is worth.

This type of analysis can be conducted by varying not only developer related parameters but also task and team parameters. For each project, the PD manager needs to decide what parameters he/she can change and observe the impact on cost and schedule. The model can also be used for process improvement purposes. The next section discusses applications relating to process improvement.

6.1.2. Process improvement

A major application of the model is the improvement of existing PD processes. There are various ways this can be achieved and some of the main applications are discussed below.

6.1.2.1. Task criticality identification

In a resource constrained PD environment, it is even more important to identify the tasks that have major impact on the PD process. In fact, Wiest (1967) describes ‘critical sequences of tasks,’ where resource availability dictates the sequence of tasks. Li and Willis (1993), as well as
Bowers (1995) further used the idea of critical task sequence to develop a time-cost tradeoff algorithm, as well as compute the level of task importance. As shown in the previous chapter (Section 5.5), task duration variability can have an impact on project completion time. While this is true for almost all of the tasks comprising a project, some tasks will have greater impact than others. Especially the tasks that are on the critical path can have significantly more impact on PD duration and cost.

Using the SimLink™ model and conducting sensitivity analysis, PD managers can identify critical tasks and observe their impact on the PD process. By optimizing resource allocation to the critical tasks, the manager can improve the overall process and ensure that the probability of delays and cost overruns is minimized.

6.1.2.2. Rework policy analysis

While iteration is fundamental aspect of PD, it also contributes significantly to the cost and schedule risk in PD projects (Browning, 1999; Osborne, 1993). Smith and Eppinger (1997b) proposed two general strategies to accelerate the PD processes. Fewer iterations can be achieved by either (a) decreasing rework probabilities \( R_{i,q}^{\text{rework}} \) or (b) absorbing rework into already planned tasks (Verganti, 1997).

Well-established coordination and communication between developers, as well as well-defined interfaces between tasks can help decrease rework probabilities (Cho & Eppinger, 2001). The SimLink™ model can be utilized to analyze the effects of varying rework probabilities on the PD process.

Another way to decrease rework probability is to absorb new task information into already existing tasks (T. Browning & Eppinger, 2002). If developers and managers can incorporate new information into already planned tasks, rework impact will not be as substantial. Otherwise, a rework task (or tasks) may be required to account for the new information. The manager can analyze the impact of flexibility in terms of incorporating rework tasks into existing task structure. As explained in Chapter 5 (subsection 5.1), one can change the 'Rework' parameters from the 'Settings' menu of the model so that new information is incorporated into tasks that are already being processed or planned.
The second strategy to accelerate the iterative process is to decrease rework impact ($K_{ij,\text{rework}}$). One way to indirectly reduce rework impact is to increase the learning curve ($L_D$) of developers, i.e., encourage developers to learn faster. Increased learning curve results in faster completed tasks, i.e., decreased rework impact. Another way to decrease rework impact is to use efficient information technology tools, as suggested by Cho and Eppinger (2001).

In summary, the SimLink™ model can be utilized to analyze both strategies of accelerating iterative PD processes, i.e., decreasing rework impact and probability. The analysis can help the managers identify more efficient rework policies to improve the PD process. Besides rework policy, increased task concurrency can also help improve PD processes. The next section describes this in more detail.

6.1.2.3. Increased task concurrency investigation

Another way to shorten the PD process is to increase task concurrency. This can be achieved by increasing availability of resources and or improving information transfer between tasks. Specifically, more resources allow scheduling multiple tasks at once (subject to task information flow constraints), which can shorten PD time but can also increase overall project cost. Improving information flow between tasks increases concurrency of tasks by allowing more task overlap, which can be accounted by adjusting the values of the overlap matrix $O(i,j)$.

6.1.3. Research-oriented investigations

While the SimLink™ simulation model has been developed to serve as a practical tool for PD managers; it can also help researchers analyze more theoretical, research-oriented problems. Three potential applications of the model to investigate different PD implementation strategies are presented next. All of these are discussed in the past literature but have been rarely used by SMEs.

6.1.3.1. Preemptive iteration

The goal of preemptive iteration is to do more iteration to yield a faster schedule. This may seem somewhat counterintuitive, however, this strategy often allows for more task concurrency. Figure 30 illustrates how preemptive iteration can compress project schedule. Specifically, without preemption, tasks $A$ and $B$ are processed sequentially and task $A$ has a duration of $T_A$, while task
$B$ has a duration of $T_{B1}$. With preemptive iteration, the duration of task $A$ does not change and while the overall time that is spent on task $B$ increases (i.e., $T_{B1} < T_{B2} + T_{B3}$), the total task completion time decreases (i.e., $T_A + T_{B1} < T_A + T_{B3}$).

![Figure 30: Effect on task duration without (left) and with (right) preemption.](image)

This simple example demonstrates how preemptive iteration can be useful in improving PD processes. Hence, the model can be utilized to investigate the effects of preemptive iteration on various process architectures. Nonetheless, it should be noted that preemptive iteration increases the cost of the project, since the time spent on task $B$ with preemption (i.e., $T_{B2} + T_{B3}$) is larger than without preemption, hence the cost is higher too. Moreover, preemptive iteration assumes that resources are available to preemptively service the tasks.

### 6.1.3.2. Follow-the-sun PD implementation

Follow-the-sun workflow arrangement assumes that the tasks are passed daily between globally distributed teams (Carmel, Dubinsky, & Espinosa, 2009). This arrangement initially started in manufacturing and nowadays is prevalent in software development (Gupta & Seshasai, 2004). The SimLink™ model is able to account for various globally distributed teams by accounting for the hours of each team ($D_h$) and developers work type ($D_t$). Also, coordination cost ($\Delta t^{CROSS}$) accounts for increased time that needs to be spent coordinating tasks between distributed PD teams.

To demonstrate the usefulness of the model in analyzing performance of distributed PD, the effects on adding two developers in different time zones are considered. Specifically, the model simulates the impact of additional developers on the mobile improvement project (presented in Section 5.4.3.1). First, it is assumed that the two developers are in the US (as in the real world scenario), however, the location of the QA has been changed to Armenia. The location of the QA
is not chosen randomly; rather the firm that manages the project has a PD team in Armenia and several QAs that can potentially join the project. Next, an additional QA has been added in Mumbai, India, where the firm wants to open an office and employ a smaller PD team. Figure 31 presents the results of the simulation. In this simulation, it is assumed that the QA in Armenia and India earn half of the hourly pay rate (i.e., 35) compared to the QA in the US. The results indicate that while distributed PD has its advantages (i.e., reducing time and cost when moving the QA to Armenia), it also has its limits. Specifically, adding a QA in India does not impact project completion time in any significant way. However, average cost increases due to additional coordination cost, which is set at 5%. Hence, SMEs should be mindful when distributing PD work across different geographical locations and can use the SimLink™ model to predict the impact of additional PD teams.

![Figure 31: Mobile development project time and cost comparison for various scenarios of distributed PD.](image)

### 6.2. Generalizability

Model generalizability is often desired but hard to implement. The reason is the fact that many models are built inductively to find a solution to a specific problem or to model a specific situation. As Größler and Milling (2007) mention, “later in the process the insights gained in the project might be generalized.” Raffo et al. (2003) define generalized simulation model as “one that can be easily adapted to multiple process contexts using significantly less effort than would
be required to develop the original model." While the SimLink™ model was tested and validates with several IT firms and a handful of projects, it can be generalized to (a) model PD processes of large enterprises (LEs), (b) hardware PD projects, and (c) aid project management efforts in other domains altogether.

6.2.1. PD processes of large enterprises

The model was originally developed to aid PD process modeling efforts in SMEs. As mentioned in Chapter 2, the PD process of SMEs and LEs is different in several aspects. Nonetheless, the structure and flexibility of the SimLink™ model allow it to account for some of the differences identified by Ghobadian and Gallear (1997). The main differences of LEs that impact the PD process are presented next, along with possible impact on modeling parameters to account for these differences.

6.2.1.1. Rigid structure and information flows, high level of formalism

Compared to LEs, SMEs generally have flat organizational structure with few layers of management. This mainly results in SMEs having more flexible structure and low level of formalism. In turn, this results in higher task coordination cost for LEs. Hence, when using the SimLink™ to model PD in LEs, the coordination cost ($\Delta S_{Cross}$) may have to be increased dramatically. In fact, there is research that suggests that communication can account up to 75% of developers' time (Hales, 1991). For this reason, the model developed by Christian (1995) may be more appropriate to model PD in LEs, where coordination cost due to high level of formalism and rigid structure is higher.

Moreover, for LEs it may be necessary to explicitly model PD managers as part of the team. Specifically, many tasks need to be approved by managers before other developers can rely on the information generated by these tasks. This can cause additional delays in processing the tasks; hence explicitly modeling this review process may be required. To account for this, an additional task processing server can be added to account for the task scheduling and coordination work that the PD manager does.
Lastly, on a positive side, rigid structure and information flow implies that task relatedness (i.e., matrices $T(i,j)$ and $O(i,j)$) are known in advance with better accuracy than in the case of an SME.

6.2.1.2. Slow response to change

Large enterprises are also slow to respond to change. This can affect the PD modeling process of SimLink™ in two main ways. First, the option to consolidate rework task with a task of the same type when that task is either in the queue waiting to be processed or already is being processed will most likely be eliminated. This is due to the fact that LEs most likely will not be able to respond and incorporate rework information into the task that has been scheduled to be processed.

6.2.1.3. Good access to human resources

Due to greater availability of resources, LEs generally have better access to human capital. This implies that resource optimization may not be as much of a challenge for LEs as it is for SMEs. Furthermore, in LEs to model a large project with SimLink™ will require explicitly modeling each developer with his/her corresponding parameters (e.g., $P_D$ and $L_D$). It may be hard to correctly indicate these values for many developers, especially given the hierarchical structure of LEs, which makes it hard to assess developer characteristics as easily and precisely as SMEs can do.

6.2.1.4. Individual creativity stifled

In LEs individual differences between developers do not matter as much as they do in SMEs. Specifically, since most developers in LEs work on pre-defined tasks using pre-specified methods, their individual characteristics do not influence project outcome as much. In short, developers act as task processing servers with similar parameters. In this case, it may be better to model one ‘average’ developer for each task type and replicate them as many times as needed to match the number of developers working on a project. It is also for this reason that previous PD models (e.g., Nicholas et al., 2011) did not explicitly model various developers; rather one common resource pool was specified from which tasks consume a certain amount of resources.
While the differences indicated above were identified in the past literature, one should keep in mind that each organization is unique and when modeling PD processes of LEs important PD factors need to be considered and incorporated into the SimLink™ model. If the model cannot account for these differences, it is better to not use the model at all then use it to make unsound recommendations.

6.2.2. Hardware PD modeling

The validation of the SimLink™ model was mainly tested on software development projects. The main reason for this was accessibility of PD data on software projects, as well as generally short PD lifecycle compared to hardware projects. Nonetheless, there are also some differences between hardware PD projects and software projects. As Leonard-Barton (1992) summarizes it, “hardware development flows through well-worn channels, but much less knowledge exists about creating application software.” Another major difference is the fact that artifacts in software projects are not as visible or well understood as in the case of hardware projects. For this reason, it is usually much easier to track progress of hardware projects. Furthermore, hardware projects are better planned, while software projects can be much more speculative and uncertain. In fact, PD managers mention that “replanning a software project is not a formality, it is required” (Wideman, 2004). It is also out of necessity that hardware projects are better planned. Specifically, hardware projects are generally more resource intensive and poor planning and/or occasional replanning can not only significantly delay the project but can cause the overall project to fail.

For the above-mentioned reasons, it is usually easier to find appropriate input values for hardware development projects. In fact, when a project plan has been completed, the PD manager will most likely have all the necessary inputs for the SimLink™ model. The downside of modeling hardware projects is the fact that most of the projects have long development cycles, hence model predictions can be for months in the future (rather than days and weeks for most software projects undertaken by SMEs). This makes it hard to validate the model using data from hardware PD projects. Nonetheless, the model is able to capture the attributes of hardware development projects and predict developmental time and cost.
6.2.3. Other domains

The structure of the SimLink™ model is such that it has the ability to model job-shop problems. Specifically, a job-shop problem is concerned with scheduling \( n \) jobs (tasks) of varying length on \( m \) identical machines in such a way that overall job completion time is minimized. The job-shop problem, in turn, is the generalized version of the traveling salesperson problem, since traveling salesperson problem is equivalent to the job-shop problem with only one machine (i.e., \( m = 1 \)). Furthermore, since the traveling salesperson problem is NP-hard, the job-shop problem is also NP-hard. The SimLink™ model can be considered as having \( n \) jobs (with stochastic durations) that need to be scheduled on \( m \) machines with (potentially) distinct performance parameters. Because an optimal solution to such a problem cannot be found (for the general case), the model simulates various scenarios and presents the outcomes.

This type of approach can be useful in a variety of different domains. For example, the model can be used to plan manufacturing operations and estimate completion time. In fact, any project that involves several machines (or individuals) processing different jobs (or tasks) can make use of the model, if appropriate input parameters can be estimated with sufficient accuracy. However, one should keep in mind that the SimLink™ model was developed specifically for PD purposes, hence the input parameters are important in modeling PD but may not be as important in other domains. Furthermore, to accurately capture processes in other domains it may be necessary to capture a different set of attributes, which the SimLink™ model does not account for.

6.3. Model Limitations

As any model, the SimLink™ simulation model also has a number of limitations, which have been identified during the development and validation process. Some of the limitations stem from the choice of the simulation paradigm and some relate to the modeling assumptions, as well as usability of the simulation application. The limitations can be grouped into three categories: a) model simplifications, b) input data availability, and 3) trust in the simulation model. These limitations are discussed further in the next subsections.
6.3.1. Model simplifications

To develop a computational model of PD, the complex PD process needs to be simplified so it can be represented in a computerized model. This entails making simplifications about the PD process to capture main PD attributes, while still being useful and computationally tractable. Logically, many attributes that impact the PD process are left out and not accounted for in the model. For example, when modeling developers the SimLink™ model captures developer performance rate and learning curve values. It is clear that these two values alone cannot capture all of the attributes that describe a developer and his/her work related attributes. It is also understandable that these values cannot be static for each developer and change over time based on a variety of factors. Nonetheless, simplifications like this are necessary to build a practical and useful model.

6.3.2. Input data availability

Even with model simplifications and assumptions, the SimLink™ requires data on tasks, developers, and teams to simulate a PD project. For example, to capture task attributes, task durations, information flow between tasks, and rework probability and impact are captured. While the previous chapter presented several case studies with sufficient input data to run the model and make useful observations and conclusions, it can be time consuming to collect necessary input data. Nevertheless, the amount of collected data is inversely proportional to the number of simplifying assumptions that are made to build the model. To be more precise, a model that has many simplifying assumptions will most likely require fewer data points. However, the accuracy and utility of such a model will also be low.

It should be noted, however, that some PD teams use project management tools that also capture a subset of parameters (e.g., likely task duration estimates, task assignments to specific developers) that SimLink™ uses. Since this data already exists, it can be used to populate the SimLink™ model. Furthermore, once the data for a specific project has been captured, it can be saved and modified if any of the project parameters change.

6.3.3. Adoption challenges

Another limitation of the model has to do with PD managers’ trust in guiding decision making using the model. As mentioned in previous literature (e.g., Smith & Morrow, 1999), many
models fail to become popular PD decision aid tools because they lack transparency and do not instill confidence in the modeling results. As with any new tool, the SimLink™ also faces adoption challenges. To mitigate PD managers’ reservations regarding the modeling results, several steps are taken. First, the operation of the model is clearly and concisely explained to the managers, so they will not take the model as a ‘black box’ that manipulates data and generates outputs. Next, previous case study results are presented to build confidence in the model. Also, sample project data is used to populate the model with the PD manager and analyze various developer and process architecture configurations. Once the usefulness of the model becomes clear to the PD managers, along with its functionality and limitations, the managers become more enthusiastic about using the model.

6.4. Summary

This chapter presented various uses of the SimLink™ model. First, the model was utilized to set cost and schedule targets and evaluate the risk of not completing the project on time. To demonstrate this, two applications of the model on estimating the risk of exceeding a certain schedule target and evaluating cost and duration tradeoff were explored. Next, various process improvement applications of the model were discussed. Specifically, the chapter considered identifying task criticality, analyzing rework policy, and increasing task concurrency to improve PD processes. Lastly, research-oriented investigations using the model were presented. First, the idea of preemptive iteration could be tested using the model. Second, follow-the-sun PD implementation was explored using data from a real world project. The results showed that distributed PD can help reduce both development time and cost, however, it also had its limits.

This chapter also analyzed generalizability of the SimLink™ model to PD processes of LEs, hardware PD modeling, and other domains that include job-shop style problems.

Next, the limitations of the model were presented. Specifically, simplifying assumptions of the model, input data availability, and adoption challenges were identified as limitations of the model.
7. Conclusions

A product development (PD) project involves numerous tasks with multiple developers working to complete these tasks. As complexity of the project increases, it becomes harder to manage the PD process and evaluate the impact of different variables on project completion time and cost. While in the past various models and frameworks were proposed to help PD managers guide and evaluate projects, these models did not capture all of the PD relevant parameters. To alleviate this drawback, in the recent years various researchers used simulation modeling techniques to develop tools for PD analysis. These tools proved to be useful in estimating PD time and cost, as well as serving as experimentation instruments to evaluate the impact of different strategies before implementing them. However, past models, in general, were developed for Large Enterprises (LEs) and PD process models and/or tools for Small and Medium Enterprises (SMEs) are limited, despite the fact that PD processes of LEs and SMEs can differ in significant ways. Moreover, past modeling literature did not account for geographically distributed PD, which has become increasingly popular in the last years both among LEs and SMEs. This has created additional challenges for PD managers of SMEs who spend significant amount of time to coordinate and optimize tasks and people.

Hence, the SimLink™ simulation model was developed for PD managers of SMEs to estimate PD project cost and completion times. Also, the tool is used to assess the impact of different attributes (e.g., resources, process flexibility, and work hours) on PD performance. This chapter summarizes the development process of the SimLink™ model, starting from identifying important modeling attributes, to the structure of the model, its implementation and applications. Also, this chapter presents key contributions of this thesis and proposes potential future work.

7.1. Modeling PD Processes in SMEs

Since PD processes of SMEs are different from PD processes of LEs, it is important to analyze important factors in PD in SMEs. Also, since distributed PD is nowadays relevant to SMEs (as well as to LEs), this thesis focuses on analyzing SMEs that are involved in distributed PD. Literature survey of previous PD process models identified a gap with respect to modeling PD processes of SMEs, in general, and SMEs involved in distributed PD, specifically. However, these past models were useful for identifying key PD process modeling attributes and the advantages and disadvantages of various simulation paradigms.
To build the SimLink™ model, modeling attributes important in distributed PD of SMEs were identified. These attributes are summarized in the next section.

7.1.1. Identifying modeling attributes

To identify relevant modeling attributes, interviews were conducted with SMEs in Armenia. All of these SMEs had PD teams in Armenia and many of them were involved in distributed PD. To evaluate the interview findings, an iterative analysis of the transcripts was used. First, from primary-cycle coding thirteen key factors were identified that, generally, successful firms had, while the less successful firms did not have. As a measure of company’s success, the past and predicted future growth of the number of companies’ employees was used. Next, secondary cycle coding yielded a set of five organizational capabilities used by the successful firms. These capabilities were:

- Flexible PD
- Modular PD
- Entrepreneurship focused human capital management
- Effective and efficient customer sensemaking
- Shared identity and goals across and within PD teams

These five capabilities were further evaluated to identify key PD attributes that each capability impacted. The results of this analysis showed that that seven key attributes needed to be taken into account to successfully model PD processes of SMEs conducting distributed PD. These key attributes were:

- Task structure includes information regarding task types, breakdowns, and their durations.
- Task relatedness refers to the information flow between tasks, such as task dependencies and parallel and sequential task processing.
- Developer flexibility accounts for the range of developer skillset and work hours.
- Rework refers to the probability that a task will need to be attended more than once and the impact that this will have on the duration of the task.
• **Team structure** captures parameters of different teams working on the same project. Specifically, team structure includes information about the number of developers assigned to each team, along with corresponding skillsets.

• **Team coordination** refers to the extra coordination effort that needs to be expanded when passing tasks from one team to another.

• **Learning/performance** refers to the reduction of task duration from working on the same task (or task type) more than once and individual performance variability of developers.

These attributes were used for developing the SimLink™ model, such that the model fully captures characteristics of the attributes. The structure of the model and its main constructs are summarized in the next section.

### 7.1.2. SimLink™ model

After reviewing existing PD process models, it became clear that none of the existing models fully captured PD attributes presented above. Hence, a new model, named SimLink™ was developed based on the discrete-event simulation (DES) paradigm to capture the effects of the PD attributes. Modeling variables corresponding to each PD attribute were identified and incorporated into the DES structure of the model. The model had three distinct parts (initial task model, rework task model, and developer model), which were accounted for by using the following queuing-based constructs: events, arrival processes, service processes, and queuing policy. The model was initially implemented in Simulink/MATLAB® environment with limited capability and later expanded by developing standalone applications for iOSTM devices and Mac OS® operating system.

#### 7.1.2.1. Model inputs and outputs

By mapping PD attributes to the DES and related queuing constructs, the SimLink™ model captures the initial tasks, rework tasks, and developer characteristics as model inputs.

• **Initial task model** represents the initial tasks that are part of a certain PD project. These tasks are specified along with projected duration and cost estimates for each task, type of the tasks, information flow between the tasks, the level of overlap (concurrency) between
the tasks, as well as task coordination cost when passing tasks between geographically distributed PD teams.

- **Rework task model** captures the fundamental, iterative aspect of PD. Specifically, the model accounts for the probability that a certain task will cause rework for another task and the impact that the rework will have.

- **Developer model** captures characteristics of PD teams and developers working on the PD project. These characteristics are listed below.
  - Number of PD teams and number of developers per team.
  - Developer work hours and type of work they can do.
  - Productivity levels and learning curves of developers.
  - Hourly pay rates for each developer.

Together these characteristics of developers allow the model to take into account each developer with sufficient accuracy to observe the impact of developer variability on the PD process, while at the same time not requiring too much/detailed information that is hard to find.

The model captures a variety of metrics useful for analyzing performance of PD processes. These metrics are divided into two groups: DES-based metrics and PD specific metrics.

- **DES based metrics** are inherently related to the event-based nature of the DES modeling paradigm. These metrics are the following:
  - Total and average wait time tasks spend waiting in the queue.
  - Total and time-average number of tasks in the queue.
  - Average utilization (i.e., percent busy time) of developers, as well as utilization of each individual developer.

These metrics are critical for evaluating PD processes; especially when one needs to optimize certain features, such as reduce the backlog of tasks (total number of tasks in the queue).

- **PD specific metrics** are the total time and cost to complete PD projects. It should be noted that due to the simulation based nature of the SimLink™ model, probability distribution functions (PDFs) are generated for project completion time and cost, as well as expected average, minimum, maximum, and standard deviation (SD) values.
Managers use PD specific metrics more often, since they provide higher level information about PD project performance. DES based metrics, on the other hand, are more useful when debugging the simulation model and analyzing certain situations (e.g., busyness level of a particular developer).

### 7.1.2.2. SimLink™ implementation

To test the simulation model, it was implemented as a software application. First, a Simulink/MATLAB® based prototype application was developed. This was used to test the feasibility of using the DES paradigm to model multiple developers working on a PD project. The MATLAB environment provided the flexibility to build the basic version of the model without committing significant resources, while providing an opportunity to analyze the main characteristics of the model.

Next, the model was expanded to include all of the identified PD attributes. Due to the limited functionality of the Simulink/MATLAB® environment and difficulties in distributing and using the model, the SimLink™ model was developed from scratch using Objective C programming language. The new modeling application greatly improved practicality and provided PD managers a user-friendly tool to study and improve PD processes (running on Mac OS©). To further improve usability and continuity of operation, a mobile version of the application was developed for Apple iPad® and iPhone® platforms. Project related SimLink™ files were synchronized through a third party file hosting service to allow seamless synchronization capabilities.

### 7.1.2.3. Model benefits

The model addresses several gaps regarding PD process modeling. First, the model provides a holistic approach in modeling PD processes of SMEs. Specifically, all of the PD attributes identified in Chapter 2 are captured in the model. These allows PD managers to evaluate more complicated situations than before and observe the impact of various variables on the performance of PD projects.

Next, the model is relevant to practice. Unlike many previous models that were developed solely in an academic setting, the SimLink™ model was developed in collaboration with SMEs and
their feedback was used to not only guide the choice of modeling variables but also for reducing the model to practice and testing the model in various scenarios.

Third, the model explicitly captures human resource availability when simulating PD processes. This is important especially when modeling PD in SMEs, since each developer can have a much greater impact on the project’s outcome compared to developers in LEs. Also, by explicitly taking into account resource constraints, managers can easily observe the gaps in availability of resources and quickly work to fill in the gaps.

Fourth, the model allows for project level optimization of PD processes in SMEs. Specifically, rather than optimizing task sequences to minimize project completion time, the model can be used to analyze the impact of several PD related attributes (e.g., task durations, developer performance ratings, rework probabilities) altogether and choose the best scenario that optimizes resource utilization, project completion time/cost.

Lastly, the model is easy to use, store and integrate with ongoing PD projects, which were all identified in the past literature as barriers for creating useful PD process models. Also, an easy-to-use graphical interface conceals most of the complexities of the model and serves as an intuitive medium to input PD related information into and observe the simulation results. This encourages model adoption and promotes its use as a PD decision aid tool.

7.1.3. Model confidence

Using multistage validation method, the model was tested on a historical data set, as well as on real world project data sets gathered from SMEs. Confidence building was an iterative process that started from verifying that the model was constructed correctly and then validating its replicative and predictive capabilities. Next, sensitivity of the output values to the changes in input parameters were studied and variability in model predictions were discussed.

The conclusions of the confidence building analysis are broken down into those that relate to model accuracy and robustness.

7.1.3.1. Model accuracy

First, the SimLink™ model was able to replicate the results of a previously developed simpler model. Specifically, both the prototype version of the model built on Simulink/MATLAB®
environment and the more advanced version were able to closely replicate the predicted completion time of an Unmanned Aerial Vehicle (UAV) project. Not only the average completion time closely matched that of the previously developed model, but also the SD values and PDFs matched the previous results.

Second, the model replicated project completion times for two software projects. The first project was a mobile software development project to support growing demands of an SME’s corporate clients. Four developers were involved in the project – two of the developers were located in Armenia and the other two in the US. The PD manager provided the work hours for each team, as well as developer characteristics (e.g., performance and learning ratings) and task characteristics (e.g., expected durations, dependencies). The model result for project completion time was close to the actual completion time value (within one SD), however, the projected value was about 10% higher than the observed value, which was due to greater task concurrency than originally expected.

The second project was a software network improvement project. Five developers distributed across two geographical locations worked on 16 tasks that comprised the project. The result of the model for project completion time was within 5% of actual project completion time.

These two examples demonstrated the model’s ability to successfully replicate results of real world SME projects. This positioned the model to be used to guide decision making in real world situation.

Lastly, the SimLink™ provided a useful insight into the bottlenecks of a real world project and helped the PD manager make an informed decision based on the model predictions. Specifically, the model predicted the project completion time and associated cost for a PD project. Initial prediction results suggested that the project might take longer to complete than anticipated by the PD manager. After analyzing cost and schedule predictions for various scenarios, the firm’s management decided to add a developer to accelerate the PD process. After completing the project and calculating total completion time it was found that the model prediction was very close to the actual project duration.
The examples provided above demonstrate the model’s ability to accurately capture PD characteristics and predict completion time, as well as help the management team guide the decision making process.

7.1.3.2. Model robustness

To test robustness of the model, SimLink™ results were analyzed for various input conditions. Since it is impossible to test the model for all input conditions, one of the projects used for validation purposes was chosen to conduct sensitivity analysis on. Sensitivity analysis is important to ensure that the model is not too sensitive to errors in input estimates. For this reason, task duration estimates, developer performance ratings, and learning curve factors were varied and the resultant changes in project completion time and cost were observed. In general, the model was robust to changes in task duration estimates and developer performance ratings. However, the model was sensitive to changes in learning curve estimates. This was due to the fact that the project under study had many tasks of the same type that developers had to process. Hence, over time the projected duration of the tasks would decrease because the developers would “learn” to complete the tasks faster. The results also indicate that, in general, sensitivity needs to be examined for each project, since different project characteristics can make the model sensitive to different parameters. Nonetheless, sensitivity of the model to the observed parameter is expected and logical and indicates that accurate estimation of the learning curve is more important than, for example, developer performance ratings (for the specific case discussed above).

Next, variability in the model output due to internal model stochasticity was analyzed. The tests showed that the model consistently generated similar output values for various simulations runs. This is important, since large differences in output values for each simulation run will greatly lower users’ trust in the model.

Also, by comparing SD values of project completion time for a UAV project, it was found that SimLink’s™ predictions were in line with previous studies. Hence, the SimLink™ is robust enough to make accurate predictions that can be used by PD managers.
7.1.4. Model limitations

It is critical to understand limitations of the SimLink™ model to correctly interpret its results and make informed decisions. First, it should be emphasized that SimLink™ is not an optimization model/tool rather it is a performance evaluation model to test the impact of various attributes on PD performance and choose the one that best fits the company’s needs. Next the following limitations of the model should be recognized.

- **Limitations due to simplifications** entail the assumptions that were made to construct a computerized model of the PD process. These include simplifications on how developers’ are represented (e.g., average values for performance ratings and learning curves representing each developer), as well as how tasks are modeled (e.g., fixed rework probability that do not depend on the developer processing the tasks). Another limiting factor is the decision to choose the DES paradigm to model the PD process. While PD processes often have discrete nature (i.e., the state of the development changes when a specific event occurs, such as completion of a task), developer characteristics, for example, change continuously rather than at discrete points in time. Hence, the choice of the paradigm limits how realistic the model can be.

- **Input data availability** can be another limiting factor. Specifically, data on tasks, developers and teams is required to run the model, which in some circumstances can be challenging to acquire, which can greatly decrease utility of the model.

- **Adoption challenges** have also been identified as a limiting factor. To be more precise, it can be hard to gain the management’s trust when using the model the first time. It is especially hard to convince the management to trust predictions of the model to make real world decisions. However, the accuracy of model predictions and ability to evaluate different scenarios can help the management over time to better understand the value of the model and use it as a routine tool to aid in PD decision making process.

In summary, while the SimLink™ has several limitations, it does not decrease the value of the model. It is the acknowledgment of these limitations that allows the PD manager to correctly use the model and get meaningful results.
7.2. Model Applications

Chapter 5 presented several uses of the model to demonstrate its validity. These examples included replicating project durations, analyzing cost vs. schedule tradeoff, and identifying a PD scenario that reduced both PD completion time and cost. In general, three key areas for model applications were identified during the development and validation process, which are discussed below.

- **Setting cost and schedule targets** is one of the main potential applications of the model. Since it is possible to analyze the tradeoff between PD schedule and cost, PD managers can evaluate different scenarios and choose one that has an acceptable risk level of schedule overrun while keeping the costs at an appropriate level. The SimLink™ model can be used to analyze only those PD scenarios that are feasible, rather than analyzing overall cost-schedule tradespace. For example, knowing how many developers can potentially work on a project, the SimLink™ should be utilized to evaluate only the scenarios that involve these developers, rather than some theoretical cost-schedule tradespace that includes imaginary developers.

- **Process improvement** is another potential application area. Specifically, the model can help identify critical tasks that have significant impact on the PD process. By paying more attention to these tasks and allocating sufficient resources, the PD manager can improve overall PD process. Other ways to improve the PD process is to analyze rework policy and reduce the probability and/or impact of rework, as well as increase concurrency of tasks by allocating more resources and/or improving information flow between tasks.

- **Research oriented evaluation** has also been identified as a potential application area of the model. There are PD strategies that have been consistently discussed in the PD literature but are usually hard to implement. The model can help quantitatively evaluate the impact of such strategies. For example, preemptive iteration has been suggested to shorten PD completion time by doing more iterations in the beginning of PD projects. While it is often risky to attempt such a strategy, the model can provide a quantitative measure of the expected reduction of PD project completion time.
Another example of a PD strategy that is hard to implement but easy to test with the model is follow-the-sun PD process implementation. Such a strategy requires setting up development activities in different time zones so that PD work can be passed from one time zone to another and effectively continues for 24 hours.

7.3. Future Work

Two main areas of future work have been identified: model related refinements / extensions and application related improvements.

First there a number of ways that the SimLink™ model can be improved to better capture PD processes in SMEs. These include:

- The current method for calculating PD project cost relies solely on hourly rates of developers. While this simplifying assumption is adequate for modeling software projects where material costs are low (or negligible), it may be inadequate for modeling hardware PD projects. Specifically, hardware projects can have significant costs associated with materials, which need to be accounted. To do this, the task model can be expanded to include a nominal cost associated with processing the task. This cost can include capital costs associated with equipment and materials that are needed to process the task. As is the case with task duration estimates, optimistic, likely, and pessimistic cost values can be specified in the model.

- The next improvement has to do with rework probabilities. The simplification that task rework probabilities are the same for all the developers can be relaxed so that the performance level of the developer is taken into account when accounting for rework probabilities. It is safe to assume that individual differences between developers (e.g., experience level, work ethic) affects the probability that a task will need rework. A relatively easy way to account for this is to tie rework probabilities to developer performance level, which is an average measure of developer’s experience, skill level, and work ethic.

- There are a few modeling variables that can be better modeled in continuous rather than in discrete domain. For example, the learning curve is one such example, which changes only after a task has been completed. However, this can be enhanced by continuously
varying the learning curve improvement based on the already completed duration of the task.

SimLink™ software application related future improvements are the following:

- The users of the SimLink™ application expressed desire to automatically update the model for an ongoing project. To be more precise, currently PD managers need to manually update the parameters of the model when tasks are completed. However, it will be more productive if the model can extract this information automatically from the PD management tool that is being used and update model predictions. This will both reduce the time spent on updating the model and will ensure that the model predictions are always up to date.
- Also, future application improvement could allow PD managers to use various PDFs when estimating task durations, instead of using a triangular distribution. One PD manager specifically asked to have the option to input beta distribution for task duration estimates, which he was more familiar with.

Although these improvements can help improve prediction accuracy and user experience, one should be mindful of the additional complexity being introduced and analyze expected benefits. Specifically, while more parameters can potentially increase predictive power of the model, they will most likely decrease practicality due to the increased demand of input data.

7.4. Contributions

The goal of the work presented in this thesis was to answer the research questions identified in Chapter 1. In achieving this goal, this thesis has contributed to the field of PD process modeling, in general, and SME process modeling, specifically. This thesis has three main contributions, which are presented below.

1. Identification of thirteen organizational factors (grouped into five categories) that successful SMEs utilize.

Past research on how SMEs develop PD capabilities is limited (Mosey, 2005). The existing literature mainly focuses on LEs (e.g., Adams-Bigelow, 2004; Leifer, 2000), despite the fact that there are numerous differences between SMEs and LEs (Ghobadian
Furthermore, SMEs and LEs do not consider the same practices to be ‘best practice’ (e.g., Nicholas et al., 2011) and Welsh (1981) notes that SMEs cannot be viewed as small versions of LEs. Hence, there is a need to evaluate SMEs to better understand success factors of PD processes.

To address this gap, this research identifies organizational factors important in PD of SMEs. These factors were identified through iterative analysis of interview transcripts and were validated to be important through firm surveys. These thirteen factors contribute to the existing literature of SME management by helping PD managers understand the factors that enable growth. These factors can be used to build better PD capabilities. Eisenhardt and Martin (2000) suggest that these factors (often referred to as routines) can be used from one development to the next, hence allowing firm managers to continuously improve their PD processes.

2. Development of a DES model that accounts for current PD practices, such as distributed and agile PD.

While there are numerous PD process models (e.g., Browning & Eppinger, 2002; Browning, 1999; Cho & Eppinger, 2001; Christian, 1995; Ford & Sterman, 1998), this work extends the capabilities of previous models to include multiple teams with different structures, additional task processing time that is required when coordinating tasks between different geographically distributed teams, as well as developer flexibility. The model is not only able to replicate results of the past (simpler) models but has been shown to accurately predict PD project completion time. The model also predicts project completion cost based on the hourly rate of developers.

3. Development of an easy-to-use decision support tool for PD managers.

Product development process model reviewers (e.g., Browning & Ramasesh, 2007; Smith & Morrow, 1999) have noted that most models lack credibility in real world and often “emphasize rigor at the potential loss of applicability” (Smith & Morrow, 1999). Moreover, most models are not implemented in a user-friendly format and have not been tested in real world environment.
To abstract complexity of the DES model away from the user, increase its applicability and credibility in real world, a PD decision support tool was developed, which encapsulated the model into a software application. The intuitive user interface of the tool and ability to run it on multiple platforms (e.g., mobile and desktop) contributes to easy deployment and operation of the tool. Using the tool, the model was validated in the real world.

Unlike past model implementations, this tool has been tested on multiple projects (both hardware and software), including replication of past results, guiding decision making in experimental and real world environments.

Next, these contributions are elaborated in more detail by addressing research questions posed earlier in this thesis. Four research questions were posed in Chapter 1. By answering the first question, “What are the important attributes and interactions that a PD process model / decision support tool needs to capture?” this thesis resulted in the:

- Identification of seven PD attributes and associated modeling variables that collectively capture PD processes in SMEs. These attributes are task structure, team structure, task relatedness, rework, developer flexibility, developer learning / performance, and task coordination cost.

- Associated with PD attributes, a set of five organizational capabilities that successful firms developed and utilized, while the less successful firms did not. These capabilities are flexible PD, modular PD, entrepreneurship focused human capital management, effective and efficient customer sensemaking, and shared identity and goals across and within PD teams.

By answering the second question, “Can the model be used to predict the impact of various PD attributes on project performance, such as project completion time and cost?” this research resulted in the:

- Building a DES model (SimLink™) that captures all of the attributes identified to impact the PD process of SMEs and verifying that the model is built correctly. Specifically, the SimLink™ model captures PD attributes at the level of granularity that has not been modeled before. Specifically, individual developers working in geographically distributed teams with different time zones are captured. Also, individual developer characteristics
and delays in coordinating tasks across geographically distributed teams are accounted for.

- Using the model to replicate project completion time results utilizing an historical data set of a hardware PD project. The results showed that the SimLink™ model successfully captures and expands capabilities of previously developed PD process models.
- Using the SimLink™ model to predict project completion time for real world projects and comparing the result to the actual findings.
- Utilizing the model to guide decision-making in real world situation by making predictions on project completion time and cost of a software project.

By answering the third question, “What level of accuracy can be expected from the model? How does it compare to real world findings?” this thesis resulted in the:

- Comparison of replicative and predictive modeling results to the real world findings. The results indicate that the model can accurately (within one SD) predict project completion time. This instills confidence in using the model as a predictive tool for PD managers to improve PD processes.
- Conducting sensitivity analysis to evaluate the impact of input errors on model outputs. The results show that in most cases the model is robust to input errors. However, the model also indicated that precisely estimating certain input variables (e.g., developer learning curve) is more important, since the model is more sensitive to estimation errors of these variables.
- Conducting stochasticity analysis to ensure that the model has an appropriate level of output variability. The analysis indicates that the output results can be replicated with great consistency, which shows that the model is consistent in its predictions. Also, predicted SD values for a hardware project are compared to SD values of previously developed and validated models. The results confirmed that the model has an appropriate level of internal stochasticity.

The above-mentioned research questions relate to the theoretical contributions of this thesis work. The next question, on the other hand, relates to the practical contribution of this research
work. Hence, by answering the last research question, “Can a moderately trained individual use the tool without requiring the use of any supporting software?” this thesis resulted in the:

- Development of a Mac OS®, iPad® and iPhone® based software application, which implements the model using an easy-to-use graphical interface. The application allows continuous operation across both mobile and Mac OS® platforms and works as a standalone application.
- Distributing the model to PD managers of SMEs to help improve their PD processes. After initially explaining how the model works, the PD managers did not have any major difficulties using the tool. Hence, a moderately trained individual can use the tool without any further support.

The complexity of PD processes has made it hard for PD managers to successfully evaluate projects and make timely and accurate decisions. This is especially applicable to SMEs conducting distributed PD, since they do not have the tools and resources that L.Es have, which makes project evaluation and optimization even more challenging. The SimLink™ model (and corresponding desktop and mobile applications) could play a significant role in helping SMEs better manage and evaluate their hardware and software PD projects in an economical way.
Appendix A: Interview Structure and Additional Findings

A.1 Main Interview Questions

Background information:

- What does your firm do (i.e., nature of work)? Please tell us a short history of your firm (including growth trajectory, product mix change over time, number of locations, etc.)
- What is the structure of the firm management?
- What is the type of firm ownership?
- What is the number of employees by level of qualification and position and what is their average salary?
- What are the approximate sales volume and profit margin?
- How did you get involved with this company?
- How long have you been employed here?

Product development:

- Please describe your new product development process in detail. More specifically, please share with us how you develop a concept idea and turn into a product/service. (The interviewee will be asked to describe the process step by step).
- What project/product management tools do you use?
- Do you use any agile product development practices?
- Does the firm collaborate with other organization/institutions? If so, please explain more.
- What are the main challenges that you encounter and how do you tackle these challenges?

Culture:

- What is it like to work at your firm? Please compare and contrast to other places you have worked at.
- Please describe what a typical workday is like.
- How much do essential employees/staff/management collaborate?
A.2 Results of Primary Cycle Interview Coding

Thirteen key factors are identified from interview transcript analysis. These factors are described next, along with examples from interviews.

1. Overlapping PD stages with short milestones was mentioned by five out of the ten firms interviewed. In fact, this was so important for some of the firms that they would not start a task unless it could be completed in less than two days. For instance, an interviewee from firm 2 mentioned,

   "The tasks are broken down into pieces that are small enough that their expected duration can be predicted with sufficient accuracy."

   Another interviewee noted,

   "[Having short milestones] helps minimize the risks and quickly understand whether something is going to work well or not at all."

   Overlapping PD stages helped conduct several iterations of the product (or product features) during a short period of time. Specifically, it was possible to get feedback on the product, while it was still being developed, rather than wait for the development phase to conclude to gather feedback. As one interviewee noted,

   "High level [product] timeline heavily relies on user feedback. When a set of features is implemented, a user experience study is conducted before the next set of feature is agreed upon."

   Hence, overlapping PD stages also allowed customer feedback to flow back easily and affect PD process.

2. Greater developer freedom is another factor that was identified from interviews to impact the PD process. Developer freedom in this context refers to developers’ ability to work flexible work hours and be creative when deciding how to approach a problem/task. Regarding flexible work hours, many of the successful firms did not have fixed work hours, i.e., developers were allowed to work different hours and not necessarily at company’s office location. In fact, a developer from firm 2 mentioned,
If there is a need to work from somewhere else [other than the company’s office], it is completely OK.

Another interviewee, founder of firm 6, mentioned,

Many employees work at odd hours from home, which lowers company’s costs – the office space only fits about half of total employees and costs less.

Furthermore, generally successful firms allowed employees to be creative when solving problems, rather than follow a prescribed ‘script.’ A developer noted,

This [flexibility to solve problems] is very important – if you reduce developers’ flexibility and do not let them really think, they will get tired of the work really soon.

Hence, flexibility to be creative and not being tied to a certain work office and/or being constrained by work hours is important for the PD process. However, must be noted that the quotes mentioned above are from employees who work on developing software. For other type of development, flexible work hours and/or location requirements may actually lower PD performance.

3. Flexible workforce skills refers to developers’ ability to work on different types of problems. For instance, a vision algorithm developer can also help design a high-speed data center. The co-founder of firm 1 mentioned,

Engineers have fluid competencies. They often work on the same project and are able to substitute each other, if needed.

Another interviewee from firm 3 noted,

Core group of engineers is willing to take on additional workload to work on others’ tasks, even if it’s outside of their area of specialization.

Flexibility in processing different tasks can greatly impact project completion time and cost, based on skill level of developers and their hourly rates.
4. Modular architecture was emphasized by several firms as the main way to plan and organize new PD activities. For many firms, modularity enabled them to not only have structured PD process but also reduce their development costs. The PD manager of firm 2 noted,

*The modular architecture, although expensive to implement in the beginning, made the rest of the development cheaper.*

Firm 8, on the other hand, uses mainly non-modular PD process. In fact, one of the engineers mentioned,

*The architecture of our products is known only after the product has been built and tested.*

More specifically, developers started working on a project knowing only high-level goals and objectives. However, costly redesigns were needed because in later development stages they discovered that product functionality was worse than originally anticipated. Modular PD architecture also requires well-specified interfaces between different product modules. This helps the team create different parallel works stream and accelerate PD process.

5. Parallel task processing was another key factor emphasized during the interviews. As one can imagine, in an ideal world, all developers would work in parallel to complete the tasks and the project faster (and potentially cheaper). However, as the co-founder of firm 3 mentioned,

*Parallel task processing is critical to shorten the development time, however, it is often necessary to wait for well-defined inputs before starting a task. Otherwise, parallel task processing can add to the development time instead.*

Parallel task processing can be even more important for firms conducting distributed PD. As a developer from firm 1 mentioned,

*It’s great when I can work on the same project with a developer from the US in parallel and get the work done in about half of the time.*

Hence, parallel task processing is an important factor in distributed PD, however, one should be able to coordinate well and be mindful of the drawbacks (e.g., more iterations).
6. *Hiring criteria based on desire and ability to learn* rather than based on the experience level and/or asking salary of prospective employees was practiced by several firms. During an interview, the co-founder of firm 3 stated,

> It is much more important to have developers who are passionate about the [product’s] idea and are willing to learn, challenge themselves and grow, as opposed to having ‘genius’ developers who are just smart.

Other firms also had similar perspectives on hiring employees. Firm 4, for example, hired several interns, paid them well and trained them for several months. The ones who demonstrated ability to learn quickly were offered full time positions.

This strategy of successful firms is in stark contrast to how less successful firms hired developers. In an interview, a manager from firm 10 noted,

> Finding employees is not an issue for us – we will pay just a little more than the average salary and will always have young people who will want to work for us.

It is, after all, not surprising that employee turnover rate at firm 10 was significantly higher compared to other interviewed firms.

7. *Creating opportunities for professional growth* was important for many firms. In fact, some firms went to great lengths to help employees grow professionally by sacrificing overall PD performance. For instance, firm 2 had the opportunity and resources to hire more people to help with the development process, but did not do it because “it helped existing developers get more cross-functional experience.”

More professional opportunities also encourage developers to take on more complex tasks. As one developer from firm 1 mentioned,

> More opportunities and having the [firm’s] management’s trust places more responsibility on you to better and with higher quality complete the project.

The same developer continued,

> While the work is more and I am busier when owning large pieces of projects, it is more fun.
Hence, professional opportunities also impact performance and learning rates of developers.

8. Greater developer availability refers to developers’ accessibility in time and space and responsiveness to the needs of the employer. Availability should also be distinguished from developer flexibility discussed earlier. Flexibility implies variability, with adaptation to changing circumstances, either internal or external to the organization (Jonsson, 2007). Whereas availability is not necessarily related to change and variation in time, it is rather a disposition or a capacity among actors that can be continuous or take different manifest forms depending on the situation (Bergman & Gardiner, 2007). Availability was often manifested by developers’ attitude to work on projects after work hours and/or on weekends. A developer from firm 5 noted,

Employees just like coming to work. Often we choose to come to work on weekends, if they are behind on work.

On contrary, employees of firms 9 and 10 waited until the official end of the working day to leave as early as they could.

9. Collecting and integrating feedback often is critical to the success of many firms. Most of the successful firms in this research meticulously collected feedback as part of their PD processes. Overlapping PD stages and short task timelines made it possible to use provided feedback to guide the development process. To make the feedback collection mechanism more effective, firms 2, 3, 4, and 5 connected developers directly with customers. In fact, firm 2 went a step further and regularly organized video conferences between lead users of their product and the PD team that was geographically away from the target market. Also, developers were allowed to drive forward certain partnerships with the clients. This approach to feedback collection was not only effective, but it also kept developers engaged and motivated.

10. Data mining was another important factor in collecting feedback from customers. In fact, firms 1 to 5 all used data mining tools of various complexities. These firms gathered data in how their products were being used and analyzed it to get insights to help improve the product. Specifically, with software products data collection tools were built in (i.e., back-end analytics). A manager from firm 2 mentioned,
It [back end analytics] is needed not only to evaluate what exists today, but also to try and predict what should be added tomorrow.

While firm 4 founder noted,

*If you skip analytics, you do not know what is happening [regarding users ability to play games].*

Despite the fact that both of these firms collected user feedback by many means, back-end analytics had become the principal tool for product improvement.

11. *Organizing focus groups* is yet another way to collect feedback from customers to help improve the product. For example, firm 2 conducted experiments by incentivizing one group of users to do a certain task and comparing the outcome with the non-incentivized group’s results. Also, the management of firm 2 organized various events (e.g., apple picking trip) for lead users to connect with each other and observe the use of the company’s product in a more natural environment. Other firms used focus groups consisting of people of certain nationalities to collect their feedback on their target markets. For instance, people born and raised in the US and Canada provided feedback, which firm 4 used to modify its product’s characteristics for the North American market.

12. *Uniting team around common goals* was identified to be a common theme for the successful firms that conducted distributed PD. One of the co-founders of firm 2 described his firm’s attitude towards uniting his team as follows:

*Everyone understands that we are working towards the same goal of building a successful product. If the product fails, then we all fail.*

Also, one of the developers from firm 1 and located in Armenia described his attitude towards his colleagues located in the US in the following way:

*We are one team and everyone is equally proficient and able to express his ideas, no matter whether they work in Armenia or in the US.*

13. *'One team' knowledge sharing culture across geographically distributed locations* is of great importance to develop and share firm knowledge. Firms 1 to 3 were involved in
distributed PD and three firms emphasized knowledge sharing across and within each PD team. The manager of firm 2 mentioned,

*Direct communication between team members in various locations is critical for promoting collaboration, sharing insights, and building team culture.*

Team 1 had weekly video conferences in which all team members from both PD locations participated.

### A.3 Routines to Build Organizational capabilities

The successful firms have also been evaluated to identify the routines (Table 29) that are employed to build the capabilities. Managers of SMEs can utilize these routines to build the necessary capabilities for their enterprises. It is important to note that these routines do not just ‘appear’ in the organization, rather leadership has to actively engage and resource teams to develop and implement them. This list is by no means comprehensive and firms will certainly differ in the ways they build PD capabilities. However, the successful firms studied in this paper (firms 1-4) had similar routines in place that allowed them to effectively build corresponding capabilities. Furthermore, in general, the less successful firms (7-10) did not have any of these routines in place. Also, the less successful firms generally exhibited the opposite routines. For example, while successful firms set short milestones, allowed flexible work hours and location requirements to promote flexible PD, the less successful firms had long milestones and set fixed work hours.

An interesting observation to make is that the top three successful firms were all engaged in distributed product development. While in the past offshoring was mainly considered as a cost saving measure for large firms, all three firms in this study engaged in distributed PD to develop new and innovative products (rather than lowering costs) by utilizing a global workforce. More specifically, firms 1 and 2 had development teams located in Armenia and in the US, while firm 3 utilized a large team in Armenia and a smaller team in Russia. Effective distributed PD has been partially enabled by advances in technology that have allowed globally distributed teams to perform knowledge work without having to always meet in person (Cummings, 2004; Humphrey, 1995; Maznevski & Chudoba, 2000).
The founders and/or managers of the successful firms have validated the capabilities and routines mentioned above. While the routines described may seem trivial and, in some cases, even simple to implement, it should be noted that organizations very often identify only with routines that conform with their current way of operations (HBS, 2013). Hence, it can take a significant amount of time to understand the value of the above-mentioned routines and use them to build organizational capabilities for PD.
Appendix B: Survey Questions and Collected Results

B.1 Survey Questions

Product Development Survey

Company name: ____________________________________________________________

Total number of employees: ______________________________________________

Person completing this survey: ____________________________________________

Date: ____________________________

Please answer the survey questions to the best of your knowledge. Your answers are confidential and individual responses will not be shared without your permission. To help answer the questions, some of the response options are elaborated after the survey questions.
1. How flexible are work hours of developers?

<table>
<thead>
<tr>
<th></th>
<th>Not at all</th>
<th>Below average flexibility</th>
<th>Average</th>
<th>Above average flexibility</th>
<th>Very flexible</th>
</tr>
</thead>
</table>

**Not at all** – developers have fixed hours they need to work (for example, 9am – 6pm)

**Average** – there are general guidelines for work hours, but developers have some leeway to choose when to come and when to leave.

**Very flexible** – developers have much freedom to choose their work hours. They can come and leave as desired, as long as they accomplish their work in a timely fashion.

2. What is the duration of average task/ticket assigned to developers?

<table>
<thead>
<tr>
<th></th>
<th>Hours</th>
<th>Day</th>
<th>Week</th>
<th>Several weeks</th>
<th>Months</th>
</tr>
</thead>
</table>

3. How often developers provide updates about their progress to the product/project manager (or to their team)? (Either formally or informally)

<table>
<thead>
<tr>
<th></th>
<th>Multiple times per day</th>
<th>Once per day</th>
<th>Multiple times per week</th>
<th>Once per week</th>
<th>Every few weeks</th>
</tr>
</thead>
</table>

4. How often do developers work outside of their main area of expertise?

<table>
<thead>
<tr>
<th></th>
<th>Never</th>
<th>Rarely</th>
<th>Occasionally</th>
<th>Often</th>
<th>Always</th>
</tr>
</thead>
</table>

**Never** – type of work that developers do never changes.

**Occasionally** – developers predominantly work in predetermined focus areas. However, in some circumstances developers may be asked to do work in an area that is outside of the scope of their specialization.
Always – developers have their areas of expertise but it is very common to work on projects outside of their area of specialization.

5. How similar is your product development process from product to product?

<table>
<thead>
<tr>
<th>Not similar</th>
<th>Below average similarity</th>
<th>Average similarity</th>
<th>Above average similarity</th>
<th>Very similar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not similar – each new product development has its own custom made process.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Somewhat similar – we generally use similar frameworks but we modify certain aspects as we see fit.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Very similar - we use the same process for developing each new product.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

6. On average, how modular is each software/product developed by your firm? (More specifically, does each individual software/product component represent a complete subunit with a well-specified interface?)

<table>
<thead>
<tr>
<th>Not modular</th>
<th>Below average modularity</th>
<th>Average modularity</th>
<th>Above average modularity</th>
<th>Very modular</th>
</tr>
</thead>
</table>

7. In general, how modular is your product development process?

<table>
<thead>
<tr>
<th>Not modular</th>
<th>Below average modularity</th>
<th>Average modularity</th>
<th>Above average modularity</th>
<th>Very modular</th>
</tr>
</thead>
</table>

8. What is the maturity of the product architecture before new product development starts?

<table>
<thead>
<tr>
<th>Not Mature</th>
<th>Below average maturity</th>
<th>Average maturity</th>
<th>Above average maturity</th>
<th>Very mature</th>
</tr>
</thead>
</table>
**Not mature** – We spend very little time on planning the architecture of a new product before starting its development.

**Average maturity** – We spend enough time planning new product architecture to ensure that we have a good general sense of what the product development process should be.

**Very mature** – We spend significant time and resources on planning new product architecture to ensure that we have a very detailed blueprint on what the product development process should be.

9. What are the hiring criteria for new employees? Specifically, what are the most important attributes for hiring a new person? Please rank the top 5 of the following (1 – most important, 5 least important).

<table>
<thead>
<tr>
<th>Rank (please fill)</th>
<th>Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Experience</td>
</tr>
<tr>
<td></td>
<td>Knowledge</td>
</tr>
<tr>
<td></td>
<td>Desire to learn</td>
</tr>
<tr>
<td></td>
<td>Ability to learn</td>
</tr>
<tr>
<td></td>
<td>Qualifications (e.g., degrees, certificates, achievements)</td>
</tr>
<tr>
<td></td>
<td>Flexibility</td>
</tr>
<tr>
<td></td>
<td>Independence</td>
</tr>
<tr>
<td></td>
<td>Attention to detail</td>
</tr>
<tr>
<td></td>
<td>Integrity</td>
</tr>
<tr>
<td></td>
<td>Decisiveness</td>
</tr>
<tr>
<td></td>
<td>Initiative-taking</td>
</tr>
<tr>
<td></td>
<td>Professionalism</td>
</tr>
<tr>
<td></td>
<td>Other, please list</td>
</tr>
</tbody>
</table>

10. How much attention is paid to helping employees develop professionally?

<table>
<thead>
<tr>
<th>Not at all</th>
<th>Below average</th>
<th>Average</th>
<th>Above average</th>
<th>Significant</th>
</tr>
</thead>
</table>

**Not at all** – employees are responsible for their own professional development.

**Average** – If we need employees to learn something to contribute to one of our projects, we’ll help them acquire the necessary skills.
Significant – We help employees grow professionally even when the newly acquired skills do not contribute directly to any of our projects or requires us to spend additional time/resources.

11. How important is customer feedback to developing new products?

<table>
<thead>
<tr>
<th>Not important</th>
<th>Below average importance</th>
<th>Average importance</th>
<th>Above average importance</th>
<th>Very important</th>
</tr>
</thead>
</table>

Not important – customer feedback is not utilized when developing new products.

Average importance – customer feedback is used occasionally when there is real need for it.

Very important – customer feedback is integral part of product development process and is collected as often as possible.

12. How often do you collect customer feedback?

Never | Rarely | Occasionally | Often | Always

13. How often rework is created due to customer feedback?

Never | Rarely | Occasionally | Often | Always

Never – customer feedback does not cause us to rework already completed tasks.

Occasionally – customer feedback causes rework occasionally when there is real need to change something.

Always – customer feedback drives product development and very often triggers rework of already completed tasks.
14. How is customer feedback collected? (Choose all that apply)

<table>
<thead>
<tr>
<th>Mark applicable methods</th>
<th>Feedback collection method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual interviews</td>
</tr>
<tr>
<td></td>
<td>Individual observations</td>
</tr>
<tr>
<td></td>
<td>Surveys</td>
</tr>
<tr>
<td></td>
<td>Focus groups</td>
</tr>
<tr>
<td></td>
<td>Reading comments/feedback provided without prompting</td>
</tr>
<tr>
<td></td>
<td>Embedded analytical techniques (such as back-end analytics)</td>
</tr>
<tr>
<td></td>
<td>Others (please specify):</td>
</tr>
</tbody>
</table>

15. How important is shared identity across geographical locations? (if applicable)

<table>
<thead>
<tr>
<th>Not important average importance</th>
<th>Below average importance</th>
<th>Average importance</th>
<th>Above average importance</th>
<th>Very important</th>
</tr>
</thead>
</table>

16. How important is for your firm’s employees to have common goals with respect to products they work on?

<table>
<thead>
<tr>
<th>Not important average importance</th>
<th>Below average importance</th>
<th>Average importance</th>
<th>Above average importance</th>
<th>Very important</th>
</tr>
</thead>
</table>

17. How important is it to have unifying culture?

<table>
<thead>
<tr>
<th>Not important average importance</th>
<th>Below average importance</th>
<th>Average importance</th>
<th>Above average importance</th>
<th>Very important</th>
</tr>
</thead>
</table>
### B.2 Collected Results

<table>
<thead>
<tr>
<th>Question</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>12</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>17</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>
Appendix C: Evaluating PDF of Triangular Distribution

Knowing the 10th and 90th percentiles of task duration, as well as the mode, it is possible to find the 0th and 100th percentiles. Let’s assume the PDF of a triangular distribution is shown below.

Using the figure above, we are given the following:

\[ S_{ADo} = S_{IEC} = 0.1 \]
\[ S_{ABC} = 1 \]
\[ \o, \p, \l \text{ are given} \]

Hence, we need to find \( A \) and \( C \), which correspond to 0th and 100th percentiles of task duration.

From the fact that \( S_{ADo} = S_{PEC} = 0.1 \), we have:

\[
\begin{cases}
axu = 0.2 \\
byv = 0.2
\end{cases}
\quad (1)
\]

where \( a = \l - \o, b = \p - \l \) and from similarities of triangles \( ADo \) and \( ABl \), \( Ao = xa \). In the same way, from similarities of triangles \( CBl \) and \( CEp \), \( pC = yb \), where \( x, y \) are coefficients.

From the fact that \( S_{ABC} = 1 \):

\[ [a(1 + x) + b(1 + y)]h = 2 \quad (2) \]

From similarities of triangles \( ADo \) and \( ABl \):

\[ u = \frac{hx}{1 + x} \quad (3) \]

From similarities of triangles \( CBl \) and \( CEp \):
Substituting (3) into (1), we have:
\[
\frac{axhx}{1 + x} = 0.2 \Rightarrow ah = \frac{0.2(1 + x)}{x^2} \tag{5}
\]

Next from the fact \(axu = byv\):
\[
\frac{axhx}{1 + x} = \frac{byhy}{1 + y} \tag{6}
\]
\[
\frac{ax^2}{1 + x} = \frac{by^2}{1 + y} \Rightarrow \frac{1 + x}{ax^2} = \frac{1 + y}{by^2} \tag{7}
\]
\[
ah(1 + x) + bh(1 + y) = 2 \Rightarrow \frac{(1 + x)^2}{x^2} + \frac{(1 + y)^2}{y^2} = 10 \tag{8}
\]

Assuming:
\[
\frac{1}{x} = z
\]
\[
\frac{1}{y} = w
\]
\[
\begin{cases}
(1 + z)^2 + (1 + w)^2 = 10 \\
\frac{1}{a}z(z + 1) = \frac{1}{b}w(1 + w)
\end{cases} \tag{9}
\]

Substituting the values of \(a\) and \(b\) back into (9):
\[
\begin{cases}
(1 + z)^2 + (1 + w)^2 = 10 \\
\frac{1}{l - a}z(z + 1) = \frac{1}{p - l}w(1 + w)
\end{cases} \tag{10}
\]

Hence, \(z\) and \(w\) can be found by solving this system (10) of two equations and two unknowns and eliminating solutions that do not satisfy \(z > 0\) and \(w > 0\) condition.
Appendix D: Description of SimLink™ Application

D.1 Screenshot of MATLAB® Prototype Application

Figure 32: MATLAB® simulation software interface.
D.2 Mac OS® SimLink™ Application

SimLink™ application on Mac OS® runs as a standalone application and can be updated easily through built-in “Check for Updates” functionality. The startup screen of the application is shown in Figure 33. From here, the user can choose to either create a new simulation project or open an existing project.

Figure 33: Mac OS® SimLink™ startup screen.

From this menu, the user also has access to recently open projects, for faster loading. Next, Figure 34 shows main window of the application. On the upper left side of the screen, the user can switch between developer, task, and timeline views (from left to right). The default view of the application shows the developers’ window. On the upper right side of the screen, the user can choose between results, settings, and run options (from left to right).
Figure 34 shows the developer view. From the bottom of the window the user can choose to add a developer, add a team, or manage the teams. Also, the user can edit existing developers’ parameters by clicking on the “tool” icon next to the developer. Furthermore, developers can be removed from a project by clicking the “x” icon.

| Developer 1 | Types | Games_back_end, api_integration |
| Developer 2 | Types | Customer_api |
| Developer 3 | Types | Api_integration |
| Developer 4 | Types | Customer_api, ads_back_end, games_back_end |

Figure 34: Developer view of the SimLink™ application.
To add and/or edit a developer, the user specifies task types, chooses the team, and enters performance rating, learning curve factors, as well as indicates hourly cost (Figure 35).

Figure 35: Add / edit developer view.

To add a team, one needs to write the name of the team, choose an appropriate color to represent in the timeline, and specify working hours (Figure 36).

To review the tasks, the user needs to click on the task icon, which changes the view of the application. In this view (Figure 37), the user can review all of the tasks and edit or remove tasks, as well as add new tasks.

Figure 36: Add team view.
Task #1
Type I games_back_end, Duration I TD (18.00, 22.00, 30.00)

Task #2
Type I customer_api, Duration I TD (5.00, 8.00, 14.00)

Task #3
Type I ads_back_end, Duration I TD (16.00, 25.00, 28.00)

Task #4
Type I games_back_end, Duration I TD (12.00, 14.00, 16.00)

Task #5
Type I api_integration, Duration I TD (2.00, 3.00, 5.00)

Task #6
Type I ads_back_end, Duration I TD (6.00, 8.00, 9.00)

Task #7
Type I games_back_end, Duration I TD (1.50, 2.00, 3.00)

Task #8
Type I api_integration, Duration I TD (15.00, 16.00, 24.00)

Task #9
Type I ads_back_end, Duration I TD (4.00, 9.00, 11.00)

Task #10
Type I customer_api, Duration I TD (18.00, 24.00, 36.00)

Task #11
Type I api_integration, Duration I TD (3.00, 5.00, 7.00)

Figure 37: Add / review tasks view of the application.

To add a new task, the user needs to specify the type of the task, its minimum duration, coordination cost, as well as optimistic, likely, and pessimistic task duration estimates Figure 38.
The user also needs to specify task relatedness (Figure 39), i.e., what other tasks the new task depends on. To indicate this, the user places check marks corresponding to the tasks that the new task depends on.

Next, the user needs to specify the overlap using the “Overlap” view (Figure 41).
Next, the user indicates rework probabilities (Figure 40) and rework impact (Figure 43) for as many iterations as needed.

Figure 41: Add task view - overlap.

Figure 40: Add task view - rework probabilities.
Before running the simulation, the user chooses appropriate setting parameters (Figure 42). From the settings view, one can choose to use iterations (or run the model just once) and specify the
number of simulation runs / iterations. Also, the user can indicate whether to include rework
tasks when the same task that the task for which rework is created is either already being
processes or in the queue waiting to be processed. Lastly, the user can specify whether
developers from different teams can pass tasks among each other and what the task scheduling
priority is (i.e., minimizing time or cost).

Lastly, to observe the results, the user can use the timeline (Figure 44) and the results view
(Figure 45).

Figure 44: Mac OS® SimLink™ application sample timeline.

Figure 45: Results for a sample project.
In case the user wants more detailed information about the simulation, “Advanced Logging” option can be turned on from the model settings. This will detail the flow of tasks through the SimLink™ model, as well as allow the user to save the log as a text file for later analysis (Figure 46).

Figure 46: Sample results using advanced logging option.
Appendix E: Support Data for Model Verification & Validation

E.1 Calculating Values for Triangular PDF

Plugging in values for $a = 10, l = 15, p = 25$ into Eq. 10, Appendix C, we get a set of equations:

\[
\begin{align*}
(1 + z)^2 + (1 + w)^2 &= 10 \\
\frac{1}{5} z(z + 1) &= \frac{1}{10} w(1 + w)
\end{align*}
\]

Taking into account the initial condition, i.e., $z > 0, w > 0$, we get the following solution:

\[
\begin{align*}
z &= 0.95 \\
w &= 1.49
\end{align*}
\]

Hence,

\[
\begin{align*}
x &= \frac{1}{0.95} = 1.05 \\
y &= \frac{1}{1.49} = 0.672
\end{align*}
\]

Therefore, the 0\textsuperscript{th} percentile of task duration is $d = 10 - 5 \times 1.05 = 4.74$ and 100\textsuperscript{th} percentile of task duration is $f = 25 + 10 \times 0.672 = 31.72$.

Knowing this and the fact that the mode of the task duration is $l = 15$, we can calculate the mean, the median, and SD.

\[
\begin{align*}
\text{Mean} &= \frac{d + l + f}{3} = \frac{4.74 + 15 + 31.72}{3} = 17.15 \\
\text{Median} &= f - \sqrt{\frac{(f - d)(f - l)}{2}} = 31.72 - \sqrt{\frac{(31.72 - 4.74)(31.72 - 15)}{2}} = 16.7 \\
\text{SD} &= \sqrt{\frac{d^2 + l^2 + f^2 - dl - df - lf}{18}} = 5.56
\end{align*}
\]
E.2 Rework Probability and Impact Matrices for UV Project

Rework probability matrix \( R_{i,q}^{\text{rework}} \) is the following:

Table 30: Rework probability matrix for UV project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.5</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.3</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td>0.5</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.4</td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.2</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.5</td>
<td>0.5</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.3</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
<td>0.4</td>
</tr>
</tbody>
</table>

Rework impact matrix \( K_{i,q}^{\text{rework}} \) is the following:

Table 31: Rework impact matrix for UV project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.4</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.1</td>
<td></td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td></td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>9</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td>0.5</td>
<td>0.5</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>0.5</td>
<td></td>
<td>0.3</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
</tr>
<tr>
<td>13</td>
<td>0.9</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>0.5</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
<td>0.8</td>
</tr>
</tbody>
</table>

\[ \text{Rework probability matrix (} R_{i,q}^{\text{rework}} \text{) is the following:} \]

\[ \text{Table 30: Rework probability matrix for UV project.} \]

\[ \text{Rework impact matrix (} K_{i,q}^{\text{rework}} \text{) is the following:} \]

\[ \text{Table 31: Rework impact matrix for UV project.} \]
It should be noted that in this example the coordination cost is zero and there is only one team working on the project. Also, rework probabilities and impact values are the same for all iterations (i.e., \( q \geq 1 \)).

**E.3 Modeling data for Mobile Development Project**

Task relatedness matrix \((T(i,j))\) for mobile development project:

*Table 32: Task relatedness matrix for mobile development project.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Rework probability matrix \((R^{i,\text{rework}}_q)\) for mobile development project, \( q = 1,2 \):

*Table 33: Rework probability matrix for mobile development project.*

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

187
Rework impact matrix \( (K_{ij}^{\text{rework}}) \) for mobile development project, \( q = 1,2 \):

Table 34: Rework impact matrix for mobile development project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>0.2</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>0.5</td>
<td>0.25</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.2</td>
<td>0.25</td>
<td>0.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
</tr>
</tbody>
</table>

E.4 Modeling Data for Network Improvement Project

Table 35: Brief task descriptions for network improvement project.

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Brief Task Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Double credit from paypal transaction</td>
</tr>
<tr>
<td>2</td>
<td>Build Android unity plug-in</td>
</tr>
<tr>
<td>3</td>
<td>Set up Mopub banner tag for Tapps</td>
</tr>
<tr>
<td>4</td>
<td>Update daily revenue analysis to catch &quot;New launched&quot; campaigns</td>
</tr>
<tr>
<td>5</td>
<td>Support {advertising id} in rich media HTML</td>
</tr>
<tr>
<td>6</td>
<td>Open access to create postbacks from Missing Transaction page for CS users</td>
</tr>
<tr>
<td>7</td>
<td>Fix Mobimeet Android postback without any transaction / action</td>
</tr>
<tr>
<td>8</td>
<td>Add Glory / CPlera non-realtime feed</td>
</tr>
<tr>
<td>9</td>
<td>Try gevent 1.0 and uwsgi async</td>
</tr>
<tr>
<td>10</td>
<td>Device entropy analysis</td>
</tr>
<tr>
<td>11</td>
<td>Support advertising ID in Android SDK</td>
</tr>
<tr>
<td>12</td>
<td>Limit per offer per publisher payouts to 100% revshare</td>
</tr>
<tr>
<td>13</td>
<td>Fix publisher_hourly_summary size</td>
</tr>
<tr>
<td>14</td>
<td>Update NewRelic agent in Gallery app</td>
</tr>
<tr>
<td>15</td>
<td>Integrate Zucks non-realtime API</td>
</tr>
<tr>
<td>16</td>
<td>Send more clear alerts which will show what was modified in changed alerts</td>
</tr>
</tbody>
</table>

Task relatedness matrix \( (T(i, j)) \) for network improvement project:
Table 36: Task relatedness matrix for network improvement project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

Rework probability matrix \( (R_{i,q}^{\text{rework}}) \) for network improvement project, \( q = 1 \). When \( q = 2 \), the values of \( R_{i,q}^{\text{rework}} \) are divided in half.

Table 37: Rework probability matrix for network improvement project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.4</td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.7</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>
Rework impact matrix \( (K_{i,q}^{j,\text{rework}}) \) for network improvement project, \( q = 1 \). When \( q = 2 \), the values of \( K_{i,q}^{j,\text{rework}} \) are divided in half.

Table 38: Rework impact matrix for network improvement project.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.2</td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>0.6</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The overlap matrix \( O(i,j) \) is zero except for the following:

Table 39: Overlap matrix for network improvement project.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>9</td>
<td>6</td>
<td></td>
<td>0.3</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td></td>
<td>0.8</td>
</tr>
</tbody>
</table>
E.5 Modeling Data for New Feature Implementation Project

This project contains short (independent) tasks that add new features to an already existing product. The reworks are generated after the QA finishes evaluating the completed tasks and submits her suggestions for improvement.

Task relatedness matrix \( T(i,j) \) for new feature implementation project is presented below. The overlap matrix \( O(i,j) \) is zero.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Rework probability matrices \( R_{iq}^{\text{rework}} \) for new feature implementation project, when \( q = 1, 2 \) are shown below.

**Table 41: Rework probability matrix for new feature implementation project (q = 1).**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0.8</td>
<td>0.5</td>
<td>0.7</td>
<td>0.5</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>0.4</td>
<td>0.6</td>
<td>0.7</td>
<td>0.7</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 42: Rework probability matrix for new feature implementation project (q = 2).**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0.4</td>
<td>0.2</td>
<td>0.3</td>
<td>0.2</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td>0.1</td>
<td>0.2</td>
<td>0.4</td>
<td>0.6</td>
<td>0.15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Rework impact matrices \( K_{i,q}^{\text{rework}} \) for new feature implementation project, when \( q = 1, 2 \) are shown below.
Table 43: Rework impact matrix for new feature implementation project \( (q = 1) \).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0.4</td>
<td>0.5</td>
<td>1</td>
<td>1</td>
<td>0.33</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>1</td>
<td>0.9</td>
<td>0.8</td>
<td>0.9</td>
<td>0.85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>0.3</td>
<td>0.3</td>
<td>0.7</td>
<td>0.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 44: Rework impact matrix for new feature implementation project \( (q = 2) \).

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
<th>18</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>0.1</td>
<td>0.2</td>
<td>0.5</td>
<td>0.5</td>
<td>0.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.3</td>
<td>0.4</td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.4</td>
<td></td>
<td></td>
<td>0.3</td>
<td>0.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

E.5 Additional Developer and QA Parameters

<table>
<thead>
<tr>
<th>Type</th>
<th>Additional developer</th>
<th>Additional QA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance rating</td>
<td>GUI, Functionality</td>
<td>QA</td>
</tr>
<tr>
<td>Cost</td>
<td>50</td>
<td>120</td>
</tr>
<tr>
<td>Learning curve</td>
<td>0.1</td>
<td>0.1</td>
</tr>
</tbody>
</table>
References


Hadjimanolis, A. (1999). Barriers to innovation for SMEs in a small less developed country (Cyprus). *Technovation, 19*(9), 561–570.


Sweetser, A. (1999). A comparison of system dynamics (SD) and discrete event simulation (DES). In *17th International Conference of the System Dynamics Society* (pp. 20–23).


